VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2013 Jaroslav Macháč

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Webově orientovaný fakturační systém
Web-based Billing System

2013 Jaroslav Macháč

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jaroslav Macháč**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Webově orientovaný fakturační systém
Web-based Billing System

Zásady pro vypracování:

Cílem práce je navrhnout a vyvinout webově - orientovanou aplikaci, která bude pokrývat procesy spojené s fakturací pro malé firmy a živnostníky.

1. Nastudujte si problematiku fakturace.
2. Sestavte přehled aktuálně využívaných webově - orientovaných Java frameworků a následně zvolte vhodný framework pro danou aplikaci.
3. Navrhněte a realizujte aplikaci pro podporu fakturace.
4. Pro perzistenci dat využijte databázový server MySql.
5. Aplikaci implementujte podle návrhového vzoru MVC (Model-View-Controller architektura).
6. Systém by měl podporovat registraci uživatelů a bezproblémov fungovat ve více uživatelském režimu.
7. Výstupní sestavy vytvořte do formátu PDF.
8. Naimplementujte modul, který bude mít za úkol zpracovávat bankovní výpisy pro potřeby aplikace.
9. Aplikace bude stabilní a bezpečná pro ochranu a zachování dat.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Gábor**

Konzultant bakalářské práce: Ing. Peter Chovanec

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlásenie

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 7. 5. 2013 .....

Touto cestou by som chcel poďakovať Ing. Michalovi Gáborovi za trpezlivosť, podporu a dobré rady pri tvorbe tejto bakalárskej práce a svojej rodine a hlavne priateľke za trpezlivosť.

Abstrakt

Táto uponáhľaná doba núti človeka čoraz viac využívať počítač, telefón a iné výdobytky modernej doby na uľahčenie a hlavne urýchlenie prace. Všimol som si to na mojom otcovi a preto sa cieľom mojej bakalárskej práce stal nastroj na fakturáciu resp. webovo-orientovaný fakturačný systém. Pokúsil som sa o skĺbenie jednoduchého ovládania a vysokej funkčnosti pre spracovanie faktúr, ich prehľad a iné funkcie, ktoré sú bližšie popísane v tejto práci. V riešení som využil ZK Framework s využitím návrhového vzoru MVC a ORM Hibernate. Samotná aplikácia je nasadená na java aplikačnom serveri napr.: Tomcat. V tejto kombinácii sa pre užívateľa naskytne pohlaď na prehľadné jednoduché UI, v ktorom je možné pracovať s dátami prostredníctvom webového prehliadača v prostredí internetu.

Kľúčové slová

ZK Framework, MVC, ORM, Hibernate, Apache TomCat, UI, fakturácia, faktúra, dodací list, Eclipse

Abstract

This hasty time forces a man in making a greater use of a computer, telephone or other moder times achievements to facilitate and mailny to speed up the work. I noticed this on my father, and therefore a tool for invoicing, or more precisely a web-oriented billing system, became the aim of my bachelor thesis. I tried combining simple operation and high funcionality for processing invoices, their overview and other functions which are described in detail in this thesis. In solution I used ZK Framework using MVC design pattern and ORM Hibernate. The application itself is applied on a Java application server e.g.: TomCat. In this combination a clear simple UI for the user is available, in which one can work with the data through a web browser on the Internet.

Keywords

ZK Framework, MVC, ORM, Hibernate, Apache TomCat, UI, invoicing, invoice, delivery note, Eclipse

Zoznam významných symbolov a skratiek

API	Application Programming Interface
ORM	Object relational mapper
HQL	Hibernate Query Language
SQL	Structured Query Language
MVC	Model-View-Controller
GWT	Google web toolkit
UI	User interface
DBMS	Database management system
JDK	Java Development Kit

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam vybraných diagramov

1. Úvod

Uponáhľaná doba núti ľudí čoraz viac tráviť svoj drahocenný v práci. Pritom zostáva rada nevyužitých technológií pre uľahčenie práce a šetrenia voľného času pre rodinu alebo relax.

Cieľom tejto bakalárskej práce je vytvoriť systém, ktorý bude šetriť čas pri fakturácií a dovolí sa zamerať užívateľovi nie na únavné mnoho krát zdĺhavé administratívne úkony ale na zdokonalovanie firmy alebo na osobný relax. Tento informačný systém je teda určený hlavne malým podnikateľom, alebo živnostníkom, ktorý nemajú radi zložité inštalácie často náročných ekonomických softvérov. Informačný systém pozostáva z troch základných častí databáza, logika a View.

Databázová časť bude mať za úlohu uchovávať a poskytovať dáta. Logika zastrešuje mnohé funkcie od riadenia View, cez výpočty až po pokyny pre Bussines vrstvu, ktorá riadi dotazovanie na databázu. View sprostredkuje užívateľovi pohľad a prácu s dátami. Pod prácou s dátami sa rozumie ich používanie pri vytváraní faktúr, dodacích listov, exportov do formátov pre tlač a náhľad, vkladanie dát, ich štatistiky a podobne.

Prvé kapitoly tejto bakalárskej práce sú skôr teoretické. Objasnená je problematika fakturácie. V ďalších kapitolách získame prehľad funkcií a výhod niektorých existujúcich riešení. Objasnenie pojmu framework. Priblíženie vybraných frameworkov ich kľúčové vlastnosti, klady a zápory. Kapitola venovaná použitým technológiám a návrhovému vzoru MVC.

V ďalších častiach sa táto práca snaží o objasnenie vlastného riešenia, ktoré bolo predmetom zadania tejto bakalárskej práce. Priblížené je fungovanie databázy a jednotlivých častí logiky. Objasnené nasadenie na aplikačný server Apache TomCat.

Súčasťou je aj užívateľská príručka, ktorá má za úlohu objasniť fungovanie aplikácie z pohľadu užívateľa. K dispozícií sú screeny obrazovky jednotlivých častí aplikácie, uložené na CD.

2. Fakturácia

2.1. Faktúra

Pri podnikaní sa podnikatelia stretávajú s rôznymi účtovnými a daňovými dokladmi. Najviac používaná je z nich je práve faktúra. Faktúru možno definovať ako účtovný doklad, ktorý zachytáva výmenný vzťah medzi dvoma subjektmi (napr. právnickou a fyzickou osobou – podnikateľom), ktorý vzniká pri výmene tovaru alebo služby. [5],[6]

Faktúra sa môže vzťahovať na:

- dodanie tovaru – v danom prípade je potrebné k faktúre pripojiť aj dodací list, ak na faktúre nie je uvedené, že faktúra je zároveň aj dodacím listom. Dodací list slúži ako doklad o potvrdení prijatia dodaného tovaru a môže aj bližšie špecifikovať dodaný tovar,
- poskytnutie služby – doklad potvrdzujúci poskytnutie služby, ktorý je obvyklé pripojiť k faktúre je výkaz o poskytnutí služieb alebo obdobný doklad o poskytnutí služieb, ktorý potvrdzuje a bližšie špecifikuje druh a rozsah poskytnutých služieb.

Faktúry je možné rozdeliť na:

- prijaté (dodávateľské faktúry) – sú to faktúry, ktoré podnikateľ prijal z externého prostredia od svojich dodávateľov alebo iných subjektov, s ktorými obchoduje,
- vydané (odberateľské faktúry) - tie podnikateľ sám vystavuje za dodanie tovaru alebo poskytnutie služby svojmu odberateľovi (zákazníkovi).

Faktúru vystavuje podnikateľ - fyzická osoba (živnostník, SZČO) alebo právnická osoba. V malých firmách vystavuje faktúru väčšinou majiteľ firmy, vo väčších firmách je to obvykle pracovník ekonomického alebo účtovného oddelenia.

Na faktúre je zo zákona povinné uviesť podpis osoby zodpovednej osoby za vniknutie skutočnosti, ktorú faktúra dokladuje. V prípade fyzickej osoby - podnikateľa to musí byť samotný podnikateľ. Ak ide o právnickú osobu, musí byť vo vnútornom predpise, tzv. internej smernici určené, kto svojím podpisom zodpovedá za vystavenie faktúr.

Podnikateľ môže fakturovať iba tie činnosti, ktoré sú predmetom jeho podnikateľskej činnosti, v ostatných prípadoch ide o neoprávnené podnikanie. V prípade, že subjekt neoprávnene podniká a to takým spôsobom, že túto činnosť vykonáva závažnejším spôsobom, tým, že použije inú ako pracovnú silu alebo získa neoprávneným podnikaním väčší prospech, môže mu byť udelený trest odňatím slobody na šesť mesiacov až tri roky.
Ak subjekt neoprávnene podniká v malom rozsahu, ale svojím konaním spôsobí značnú škodu jeho trest môže byť vo výške jeden až päť rokov odňatia slobody a keď spôsobí škodu veľkého rozsahu alebo ako člen nebezpečného zoskupenia - na štyri až osem rokov odňatia slobody.

Podnikateľ môže fakturovať iba tie činnosti, ktoré sú predmetom jeho podnikateľskej činnosti, v ostatných prípadoch ide o neoprávnené podnikanie. Je dôležité dať si pozor, najmä keď

podnikateľ podniká už niekoľko rokov a možno si celkom nepamätá, aké činnosti sú predmetom jeho podnikania, na predmet fakturácie, pretože kto neoprávnene podniká v malom rozsahu, môže byť podľa Trestného zákona potrestaný odňatím slobody až na jeden rok. V prípade, že subjekt neoprávnene podniká a to takým spôsobom, že túto činnosť vykonáva závažnejším spôsobom, tým, že použije inú ako pracovnú silu alebo získa neoprávneným podnikaním väčší prospech, môže mu byť udelený trest odňatím slobody na šesť mesiacov až tri roky.
Ak subjekt neoprávnene podniká v malom rozsahu, ale svojím konaním spôsobí značnú škodu – jeho trest môže byť vo výške jeden až päť rokov odňatia slobody a keď spôsobí škodu veľkého rozsahu alebo ako člen nebezpečného zoskupenia - na štyri až osem rokov odňatia slobody.

Ak podnikateľ prijme od svojho dodávateľa faktúru, je povinný doplniť túto faktúru o náležitosti podľa zákona o účtovníctve, ktorými musí byť:

- podpisový záznam osoby zodpovednej za jeho zaúčtovanie,
- označenie účtov, na ktorých sa účtovný prípad zaúčtuje v účtovných jednotkách účtujúcich v sústave podvojného účtovníctva, ak to nevyplýva z programového vybavenia. Ide o označenie účtovacím predpisom, tzv. predkontáciou.

Podnikateľ je povinný vyznačiť na faktúre aj zaúčtovanie faktúry a to buď priamo na faktúre alebo pripojí k faktúre predkontačný lístok, na ktorom je uvedený účtovací predpis. Súčasťou faktúry musí byť taktiež podpis osoby, ktorá faktúru zaúčtovala. Tieto činnosti si zabezpečuje, ak je to fyzická osoba - sám podnikateľ alebo externý účtovník (účtovnícka firma), ktorý vedie pre podnikateľa účtovníctvo. V prípade právnickej osoby je to väčšinou pracovník ekonomického alebo účtovného oddelenia alebo pracovník externej účtovníckej firmy, ktorá pre podnikateľa zabezpečuje vedenia účtovníctva.

2.1.1. Náležitosti faktúry

Faktúra musí spĺňať viacero náležitostí a jej obsah ovplyvňuje skutočnosť či podnikateľ je alebo nie je platiteľom dane z pridanej hodnoty. Ak podnikateľ nie je platiteľom dane z pridanej hodnoty, riadi sa náležitosťami faktúry podľa Zákona o účtovníctve. Ak podnikateľ je platiteľom dane z pridanej hodnoty, musí faktúra obsahovať náležitosti aj podľa zákona o dani z pridanej hodnoty.

2.1.1.1. Náležitosti faktúry pri dodaní tovaru a služby v tuzemsku

- meno a adresa sídla, miesta podnikania, prípadne prevádzkarne platiteľa, ktorý dodáva tovar alebo službu, a jeho identifikačné číslo pre daň,
- meno a adresa sídla, miesta podnikania, prípadne prevádzkarne alebo bydliska príjemcu tovaru alebo služby a jeho identifikačné číslo pre daň, ak mu je pridelené,

- poradové číslo faktúry,
- dátum, keď bol tovar alebo služba dodaná, alebo dátum, keď bola platba prijatá, ak tento dátum možno určiť a ak sa odlišuje od dátumu vyhotovenia faktúry,
- dátum vyhotovenia faktúry,
- množstvo a druh dodaného tovaru alebo rozsah a druh dodanej služby,
- základ dane, jednotkovú cenu bez dane a zľavy a rabaty, ak nie sú obsiahnuté v jednotkovej cene,
- uplatnenú sadzbu dane alebo údaj o oslobodení od dane,
- výšku dane spolu v eurách, ktorá sa má zaplatiť.

2.1.1.2. Podnikateľ nie je platiteľom dane z pridanej hodnoty

Faktúra v prípade, že podnikateľ nie je platiteľom dane z pridanej hodnoty, musí spĺňať náležitosti faktúry podľa § 10 zákona o účtovníctve.

Náležitosti faktúry podľa Zákona o účtovníctve:

- označenie účtovného dokladu,
- obsah účtovného prípadu a označenie jeho účastníkov,
- peňažná suma alebo údaj o cene za mernú jednotku a vyjadrenie množstva,
- dátum vyhotovenia účtovného dokladu,
- dátum uskutočnenia účtovného prípadu, ak nie je zhodný s dátumom vyhotovenia,
- podpisový záznam osoby zodpovednej za účtovný prípad v účtovnej jednotke a podpisový záznam osoby zodpovednej za jeho zaúčtovanie,
- označenie účtov, na ktorých sa účtovný prípad zaúčtuje v účtovných jednotkách účtujúcich v sústave podvojného účtovníctva, ak to nevyplýva z programového vybavenia.

Zákon o účtovníctve upresňuje, čo sa rozumie pod pojmom podpisový záznam. Je to účtovný záznam, ktorého obsahom je vlastnoručný podpis, alebo obdobný preukázateľný účtovný záznam nahrádzajúci vlastnoručný podpis v technickej forme. Obdive formy sa považujú za rovnocenné.

Faktúra č. 124/2013

poradie faktúry/rok

Dátum vystavenia faktúry:	11.11.2013
Dátum dodania :	11.11.2013
Splatnosť:	30.11.2013

Dodávateľ:		Odberateľ:	
Brusenie, s.r.o.		Kamélia, s.r.o.	
Lesná 34/11		Strieborná 9	
010 01 Žilina		974 01 Banská Bystrica	
IČO:	52 639 458	IČO:	45 789 123
DIČ:	2024444444	DIČ:	2025555555
Bankové spojenie:	4581256986/0900	IČ DPH:	SK 2025555555
Dodávateľ nie je platiteľom DPH		Dodacia adresa:	-
		Bankové spojenie:	6541278534/1100

Fakturujeme Vám dodanie tovaru:

Názov	Mer. jedn.	Množstvo	Jedn. cena (€)	Spolu (€)
Vrták	ks	10	3,42	34,2
Fréza	ks	15	5,87	88,05
Píla	ks	20	5,44	108,8

Napríklad počet kusov, metrov, litrov

Uvedie sa cena za jednu mernú jednotku

Cena celkom :	231,05
Cent.vyrovnanie :	
Celkom k úhrade :	**231,05**

Pdpis

Podpis

Prevzal: Silvia Mýtna **Vyhotovil :** Ondrej Sluš

Obrázok 1: Vzor faktúry neplatcu DPH

2.1.1.3. Podnikateľ je platiteľom dane z pridanej hodnoty

Ak je podnikateľ platiteľom dane z pridanej hodnoty, musí faktúra obsahovať náležitosti podľa zákona o účtovníctve, ale aj podľa zákona o dani z pridanej hodnoty. Náležitosti týkajúce sa faktúry v prípade podnikateľa - platiteľa dane z pridanej hodnoty upravuje Zákon o dani z pridanej hodnoty najmä v §71 až §76.

Faktúru v prípade platiteľa dane z pridanej hodnoty je možné vystaviť buď za službu alebo tovar a to najneskôr do 15 dní od vzniku daňovej povinnosti. Podnikateľ musí faktúru vystaviť aj v prípade, že prijal platbu (zálohu, preddavok) pred dodaním tovaru alebo služby, a to do 15 dní od jej prijatia.

Zákon o dani z pridanej hodnoty v § 19 definuje pojem vznik daňovej povinnosti, v tomto paragrafe je možné nájsť rôzne špecifické prípady vzniku daňovej povinnosti. Najčastejšie sa však za deň vzniku daňovej povinnosti považuje:

- **deň dodania tovaru** - dňom dodania tovaru je deň, keď kupujúci nadobudne právo nakladať s tovarom ako vlastník,
- **deň dodania služby**,
- **deň prijatia platby** - ak je platba prijatá pred dodaním tovaru alebo služby, vzniká daňová povinnosť z prijatej platby dňom prijatia platby.

Ak sa dodanie tovaru alebo služby uskutočňuje čiastkovo alebo opakovane, považuje sa tovar alebo služba za dodanú najneskôr posledným dňom obdobia, na ktoré sa platba za opakovane alebo čiastkovo dodávaný tovar alebo službu vzťahuje. Za čiastkové dodanie tovaru alebo služby sa považuje také dodanie tovaru alebo služby, ktoré predstavuje časť celkového plnenia, na ktoré je uzavretá zmluva. Za opakované dodanie tovaru alebo služby sa považuje dodanie rovnakého druhu tovaru alebo služby v opakovaných dohodnutých lehotách.

2.1.1.3.1. Zodpovednosť za správnosť údajov na faktúre.

Podľa zákona o dani z pridanej hodnoty zodpovedá za správnosť údajov vo faktúre a za včasnosť jej vyhotovenia platiteľ dane z pridanej hodnoty, ktorý dodáva tovar alebo službu, a to aj v prípade, ak je faktúra vyhotovená prostredníctvom inej osoby alebo zákazníkom.

Faktúra č. 123/2013

Dátum vystavenia faktúry:	11.11.2013
Dátum zdaniteľného plnenia:	11.11.2013
Splatnosť:	26.11.2013

poradie faktúry/rok

Dátum dodania tovaru alebo poskytnutia služby

Dodávateľ:		Odberateľ:	
Lanium, s.r.o.		Dzunglee, s.r.o.	
Povrazová 18		Hlavná 132/78	
811 02 Bratislava		911 08 Trenčín	
Obchodnom registri na Okresnom súde Bratislava I, oddiel Sro, vložka číslo 13435/B		registrácia: Spoločnosť je zapísaná v Obchodnom registri na Okresnom súde Trenčín, oddiel Sro, vložka číslo 13635/B	
IČO:	12 456 789	IČO:	98 785 321
DIČ:	2 022 222 222	DIČ:	2023333333
IČ DPH:	SK 2022222222	IČ DPH:	SK 2023333333
Bankové spojenie:	2785469645/0900	Dodacia adresa:	911 08 Trenčín
		Bankové spojenie:	3478512762/1100

Fakturujeme Vám dodanie tovaru:

Názov	Merná jedn.	Množstvo	Jedn. cena bez DPH	Spolu bez DPH (€)	Sadzba DPH (%)	DPH (€)	Spolu s DPH (€)
Vrtak	ks	15	11	165	20%	33	198
Povraznio	ks	10	8,3	83	20%	16,6	99,6

napríklad počet kusov, metrov, litrov a pod.

Cena ja jednu mernú jednotku

Cena celkom (€) :	Základ:	248	DPH:	49,6	297,6
Cent.vyrovnanie :					
Celkom k úhrade (€)					297,6

Podpis

Podpis

Prevzal:	Ján Lanius	Vyhotovil :	Petra Červienkov

Obrázok 2: Vzor faktúry platcu DPH

2.1.2. Spôsob doručenia faktúry

Faktúru je možné odoslať zákazníkovi dvoma spôsobmi:

- písomne – zaslanie prostredníctvom pošty, osobne alebo podobným spôsobom,
- elektronicky - podnikateľ (platiteľ dane z pridanej hodnoty) so súhlasom zákazníka môže faktúru sprístupniť elektronicky. Vierohodnosť pôvodu a neporušenosť obsahu faktúry poslanej alebo sprístupnenej elektronicky musí byť zaručená elektronickým podpisom. Odkaz elektronickej výmeny údajov pozostáva zo súboru dielov, štruktúrovaných na základe schválenej normy, vyhotovených vo formáte čitateľnom počítačom, ktoré môžu byť automaticky a nedvojzmyselne spracované.

3. Vybrané existujúce riešenia fakturačných systémov

Táto kapitola je venovaná už existujúcim riešeniami fakturačných systémov. Zo začiatku boli tieto systémy výhradne desktopové no postupom času ľudia kvôli hektickej dobe a honbou za bohatstvom alebo skôr v boji o prežitie začali mať potrebu prístupu k ich firemným dátam nie len z kancelárie ale aj z domu, bytu , chaty či dokonca na dovolenke. Preto sa dostávajú do popredia webovo orientované fakturačné a účtovné systémy. Pre prístup postačí nenáročný počítač web browser a pripojenie na internet. Umažeme si dve riešenia webové a desktopové, pozrieme sa na ich klady a zápory.

3.1. Super Faktúra

Sú jednotkou na trhu čo sa týka webovej fakturácie, poskytujú všetky služby. Na vyber mame z dvoch balíčkov :

- základný
- prémiový

Nasledujúca tabuľkaTabuľka 1: Tabuľka funkcií SuperFaktura nám priblíži jednotlivé funkcie poskytovaných balíčkov :

	Základný	**Prémiový**
	4.90 € / mesiac	**14.90 € / mesiac**
Najvhodnejší pre	živnostníkov	malé firmy, eshopy
Faktúry	✓	✓
Náklady	✓	✓
Kontakty	✓	✓
Cenové ponuky	✓	✓
Import - Export agendy	✓	✓
Sklad	✓	✓
Odosielanie faktúr a obálok cez Hybridnú poštu	čiernobielo	farebne
Prístup pre viac používateľov		✓
Automatické odosielanie opakujúcich sa faktúr		✓
Tagovanie faktúr a nákladov		✓
API (prepojenie s e-shopom)		✓
Rôzne číselné rady pre faktúry		✓

Tabuľka 1: Tabuľka funkcií SuperFaktura

Výhody :

- žiadna potrebná inštalácia, nezáleží na tom, aký používate operačný systém. či prehliadač,
- netreba sa rozhodovať medzi rôznymi verziami, je jedna a tá najaktuálnejšia,
- dáta elektronického faktúrovania sú zabezpečené,
- je online, dostupnosť odkiaľkoľvek, kde je pripojenie na internet,
- finančné prehľady o tržbách, zisku, daniach a platobnej disciplíne sú k dispozícii v reálnom čase,
- možnosť bezproblémového importu údajov z Excelu, Pohody a iných účtovných softwarov,
- vlastný blog o novinkách systému,
- API na automatizáciu faktúrovania pre e-shopy.

Nevýhody :

- cena v rámci online fakturácie,
- miestami zmätené ovládanie,
- pocit ludi ze ich dáta môžu byt zneužité,
- nevhodný pre veľké firmy.

3.2. Pohoda ekonomický systém

Jedná sa o desktopový softvér, takže je potrebná inštalácia na priamo na počítač. Ponúka veľké množstvo funkcií no pre malého podnikateľa je zbytočne drahý a zložitý, hodí sa skôr pre väčšie firmy, ktoré zamestnávajú účtovníkov. Prakticky obyčajný človek by musel stráviť pár dní na školení aby pochopil fungovanie tohto systému. Tabuľka funkcií v tomto prípade nemá zmysel, tento systém obsahuje všetky funkcie co sa týka fakturovania, funkcionalitou prevyšuje online fakturácie. Je to spôsobené dlhou tradíciou tohto systému.

Výhody :

- celá agenda v jednom balíku,
- možná ročná uzávierka minulého obdobia bez straty dát v novom roku, pokiaľ sa už účtovalo v novom roku,
- previazanosť skladového hospodárstva (generovanie príjmu a výdaja),
- široká škála tlačových zostáv,
- homebanking, internetové obchody,
- stabilita programu (Access platforma),
- možnosť členenia skladu priamo pri príjme na sklad.

Nevýhody :

- nutnosť inštalácie,
- prístup len s desktop pc, kde je program nainštalovaný pokiaľ nemáme rozšírenú licenciu,
- cena balíčkov softvéru niekoľko násobne drahšia ako u online fakturácie,
- zložitosť celého systému,
- nutnosť školenia, ktoré je nákladne na čas ale aj financie,
- využitie plného potenciálu softvéru iba väčšie firmy.

3.3. Zhrnutie

Online účtovníctvo a online fakturácia sú ešte mladé, a v očiach nedôverčivého zákazníka nebezpečné, pritom špičkové online servre sú niekoľko násobne viac chránené z hľadiska zálohovania dát a bezpečnosti ako samotné počítače. Ich vývoj však napreduje a dostavajú sa viac do povedomia ľudí a stále pribúdajú nové funkcie. Niekoľko rokov môžme čakať kompletnú transformáciu desktopových softvérov na webové, ktorá sa už pomaly deje. Moc pristúpiť k svojim dátam z každého kúta sveta, kde sa nachádza prípojka do internetu je silná a túto skutočnosť si uvedomujú aj tvorcovia desktopových softvérov.

V porovnaní týchto dvoch typov z hľadiska funkcií, momentálne vedú desktopové a z hľadiska mobility a jednoduchosti zasa webové je len na zákazníkovi, aby sa rozhodol aká možnosť je pre neho ta pravá.

4. Frameworky

Softvérové inžinierstvo rieši rôznorodé problémy. V softvérovom inžinierstve sa vytvára široká škála informačných systémov, keďže ich je obrovské množstvo a v dnešnej dobe sa nachádzajúcu všade okolo nás tak spomeniem iba pár príkladov, ako sú napr. medicínske, strojárenské, lodné, firemné a mnohé ďalšie. Hoci je vidieť, že Softvérové inžinierstvo je späté s obrovskou oblasťou ľudských činností, časté problémy sa neustále opakujú. K ich riešenie slúžia práve frameworky.

4.1. Hlavná podstata frameworkov

Slovo framework je zložené z dvoch anglických slov Frame a Work. Frame znamená rámec. Work znamená práca. Framework je teda konštrukcie presne určená konštrukcia, ktorá rieši za programátora určitú časť nejakého softvérového problému. Napríklad ak tvoríte Informačný systém, dostupný cez web, používajú sa práve webové frameworky ako napríklad ZK Framework, ktorá uľahčuje tvorbu webového rozhrania a my sa môžeme sústrediť na podstatu vytváraného Informačného systému ďalší typ frameworku je napr. ORM Framework Hibernate, ktorý nám uľahčuje komunikáciu s databázami.

4.2. Prehľad niektorých java frameworkov pre tvorbu UI

4.2.1. Google Web Toolkit

Google web toolkit (ďalej len GWT) je kompaktný web aplikačný framework. Zameraný je na webové aplikácie s komplexným užívateľským rozhraním, ktorého logiku na strane klienta zabezpečuje JavaScript a asynchrónne volanej serverovej logiky pomocou technológie AJAX[1]. Pričom všetko je vyvíjané na JAVA platforme.[8]

4.2.1.1. Hlavné komponenty GWT

- Kompilátor Javy do JavaScriptu kľúčový komponent, v momente keď je Java kód pripravený na nasadenie prekompiluje sa do JavaScriptu, má dva módy kompilácie. V prvom prípade je výsledný JavaScript čitateľný pre človeka, to je vhodné pri testovaní. Druhý prípad, keď je kód pripravený na nasadenie do produkcie, optimalizovaný na výkon, veľkosť, špeciálne pre jednotlivé prehliadače a pre jazykové mutácie.
- Emulátor Java platformy podpora syntax, konštrukcii jazyka a baličky základnej časti API sú re-implementované v JavaScripte,
- GWT Hosted Mode ladiaco-testovací koncept, pri ktorom sa Java kód neprekladá do JavaScriptu, ale JVM[2] spúšťa kód aplikácie ako byte kód v ladiacom móde v špeciálne

1 Asynchronous JavaScript and XML – technológia interaktívneho vývoja webových aplikácií, ktorá umožňuje meniť obsah webovej stránky bez nutnosti znovu načítania.

2 Java Virtual Machine - je sada počítačových programov a dátových štruktúr, ktorá využíva modul virtuálneho stroja na spustenie programov a skriptov vytvorených v jazyku Java.

upravenom prehliadači, štandardne na zabudovanej inštancii Jetty[] web servera. Prehliadač pozostáva z dvoch okien. V prvom hlavnom okne je výpis ladenia a výnimky, možnosť reštartovania web servera a spustenia hosted prehliadaču. V druhom okne – v hosted prehliadači beží samotná webová aplikácia, ktorú takto môžeme testovať.

Hlavné výhody a nevýhody GWT:

Výhody :

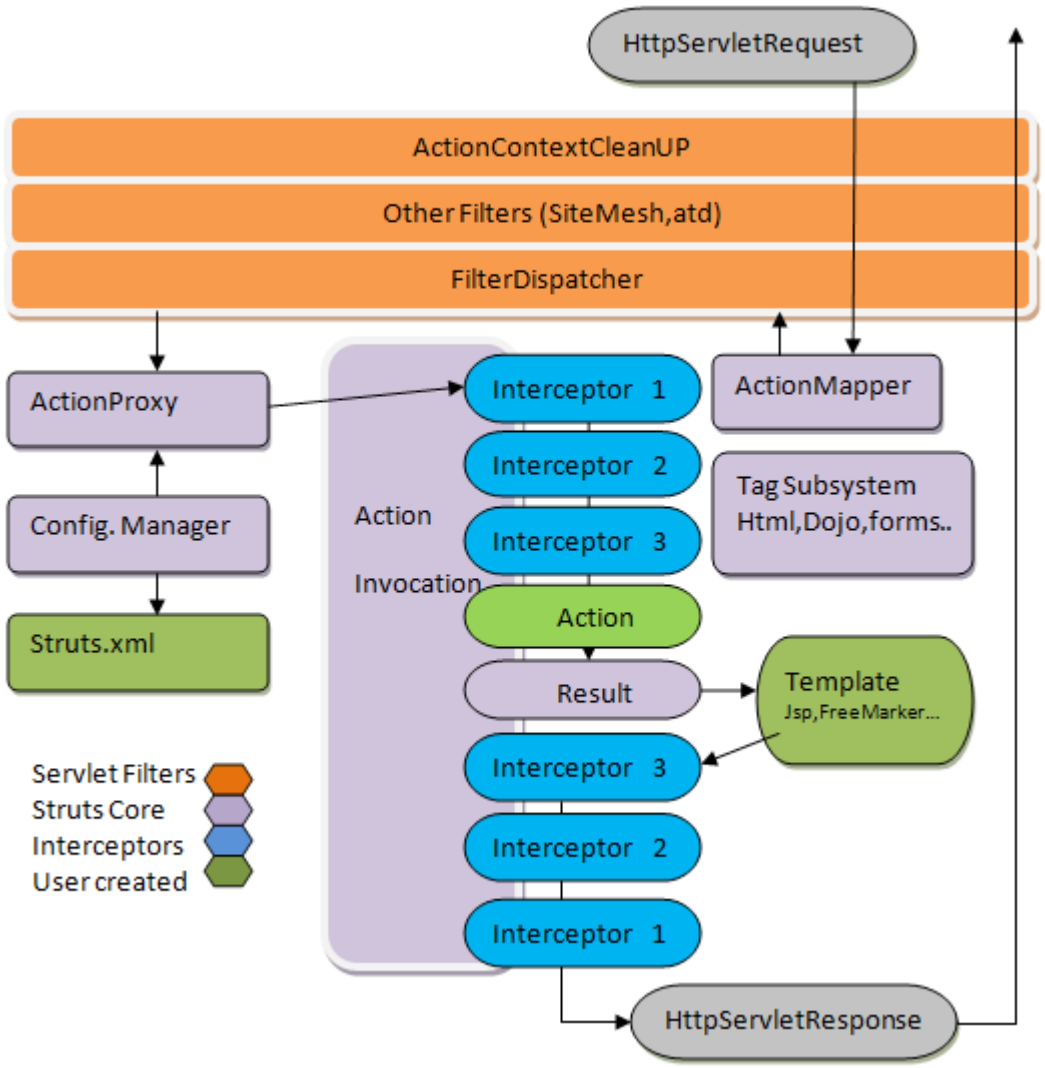
- kvalitná a prehľadná dokumentácia,
- CSS a obrázky sú vysoko optimalizované,
- refaktoring,
- jednoduchá komunikácia klient / server
- debugovanie a hot swapping,

Nevýhody :

- pomalý štart development módu,
- pre programátora neprívetivý UI dizajnér,
- spomalenie vývojového prostredia Eclipse pri editácií XML, dizajnér,
- Návrh UI – programovo :
    1. neprehľadné,
    2. Pomalé,
- Rozšírenie, úprava štýlu alebo tvorba nového widgetu je veľmi náročná a zdĺhavá

4.2.2. Struts 2

Struts je webový aplikačný framework. Jeho vývoj prebieha v rámci organizácie Apache Software Fundation, pod ich licenciou Apache Licencia 2.0. Pôvodná verzia má pomerne hlboké korene, ktoré siahajú až do roku 2000. Preto je tento framework vcelku známi nie len na poli programátorov ale aj zákazníkov. Druhý relase vyšiel v roku 2005 od predchádzajúcej verzie sa mnohé zmenilo. Napríklad Action trieda nemusí dediť ani implementovať žiadne rozhranie, pre každú HTTP požiadavku je vytvorená nová inštancia typu Action, žiadna závislosť na Servletoch a mnohé ďalšie. Na Obrázok 3: Realizácia HTTP požiadavky Struts 2 môžme vidieť realizáciu HTTP požiadavky.



Obrázok 3: Realizácia HTTP požiadavky Struts 2

Kľúčové vlastnosti a komponenty:

- pracuje výhradne s POJO[3] objektami,
- úzka spolupráca s frameworkom Spring,
- neskrýva bezstavovosť HTTP[4] protokolu,
- v niektorých prvkoch (Checkbox) podporuje stavovosť,

3 Plain old Java objects – takzvané staré objekty v Jave
4 Hypertext Transfer Protocol - je internetový protokol určený pre výmenu hypertextových dokumentov vo formáte HTML.

- podpora JSP[5], JSF[6], JTL[7], Freemarket, Velocity pro View,
- podpora JDBC[8], EJB[9], Hibernate a dalších pre Model.

Výhody :

- intuitívna práca s MVC,
- je open source,
- plugin pre známe vývojové prostredie Eclipse,
- dobrá orientácia pri tvorbe projektov,
- ponúka pomerne široké možnosti integrácie.

Nevýhody

- neprehľadná dokumentácia,
- mladosť frameworku,
- nekompletné tutoriály.

4.2.3. ZK Framework

ZK Framework(ďalej len ZK) je open-source Java Web aplikačný framework, ktorý poskytuje nielen komfortné užívateľské prostredie, ale tiež jednoduchý programovací model.[3]

Jadro ZK je na báze Ajax, je to udalosťami riadený mechanizmus s bohatou sadou XUL[10] komponentov. Programátori môžu navrhnúť ich web aplikácie, ktoré budú obsahovať množstvo XUL / XHTML[11] zložiek a manipulovať s nimi na základe udalostí vyvolaných koncovími užívateľskými aktivitami, ako bolo roky zvykom v desktopových aplikácií.

ZK má takzvaný server-centric prístup, v ktorom je synchronizácia priečinkov a udalosti medzi klientmi a servermi vykonávaná automaticky v engine a Ajax programátorské kódy sú úplne

5 Java Server Pages -je technológia Java, ktorá pomáha softvérovým vývojárom obsluhovať dynamicky generované webové stránky, založené na HTML, XML alebo ostatných typoch dokumentov.
6 JavaServer Faces - Hlavnou myšlienkou je možnosť čistejšieho vývoja profesionálnych Web aplikácií. Vývojári definujú užívateľský interface pomocou špeciálnych XML.
7 Java tool language
8 Java Database Connectivity - je API pre programátorov v programovacom jazyku Java, ktoré definuje jednotné rozhranie pre prístup k relačných databáz.
9 Enterprise Java Beans - sú riadené, serverové komponenty umožňujúce modulárnu tvorbu podnikových aplikácií.
10 Ide o formát pre tvorbu multiplatformního grafického rozhrania, ktorý je používaný v produktoch Mozilla ako napríklad Firefox či Thunderbird.
11 Extensible hypertext markup language - je značkovací jazyk pre tvorbu hypertextových dokumentov v prostredí WWW.

transparentné pre vývojárov webových aplikácií. Preto sa koncovému užívateľovi dostane interakcia a schopnosť reagovať podobne ako pri desktopovej aplikácií.

V doplnenie založeného na komponentoch programovanie v podobným spôsobom, Swing, ZK podporuje značkovací jazyk ZUL, pre bohaté definíciu užívateľského rozhrania s názvom ZUL.

Výhody

- má jednu z najlepších dokumentácií,
- tutoriály prepracovane do detailov,
- možnosť spustenia demo verzie, kde sa nachádzajú rôzne komponenty,
- reportované chyby opravuje komunita v priebehu niekoľkých hodín,
- dva typy prístupu MVC a MVVM[12],
- umožňuje sústrediť všetku logiku programovania na serveri,
- intuitívny dizajnér s možnosťou náhľadu na práve vytvárané UI,
- široká škála komponentov programátor sa môže sústrediť na vývoj a funkčnosť aplikácie, tim padom sa podstatne skracuje doba vývoja aplikácie,
- je kompatibilný s XHTML, ZUL je ľahko použiteľný pre vývojárov, ktorí poznajú HTML,
- vloženie JavaScriptu umožňuje vlastné nastavenie,
- vývojár nemusí nutne mat znalosti JavaScriptu alebo Ajaxu,
- open-source,
- dobre vyzerajúci dizajn hotovej aplikácie.

Nevýhody :

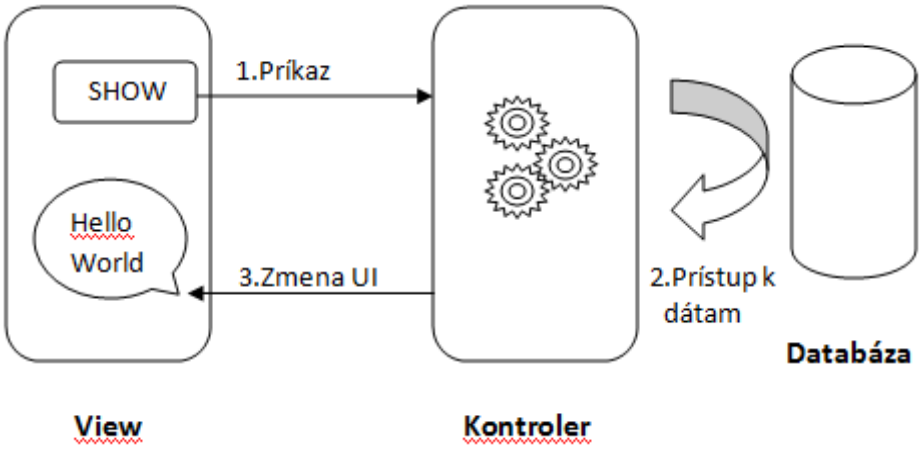
- detailná upráva správania sa dizajnu je problémová,
- nekompatibilita medzi rôznymi browsermi**,**
- nepodporuje offline mód,

12 Model-View-ViewModel – obdoba MVC

5. Technológie a návrhové vzory

5.1. MVC

Model-View-Controller (MVC) má svoje začiatky má v Smalltalku, kde sa pôvod nepoužíval na zakomponovanie tradičného postupu vstup–spracovanie–výstup do UI programov. Nemenej vhodný je ale aj pre použitie vo viacvrstvových webových aplikáciách. Ako vidíme na Obrázok 4:MVC presne oddeľuje dáta, bussiness logiku a užívateľské rozhranie



Obrázok 4:MVC presne oddeľuje dáta, bussiness logiku a užívateľské rozhranie

5.1.1. Model

Je dátovým ale aj funkčným základom aplikácie. Ponúkne prostriedky nie len pre prístup k databáze a stavom aplikácie, ale aj pre ich zmenu aktualizáciu a ukladanie. Je to vlastne softvérový obraz reálneho nášho sveta. Ako celok by mal byť v aplikácii zapuzdrený a pre View a Controller ponúkať rozhranie, ktoré musí byť presne zadefinované. Pre implementáciu sa najčastejšie je pomocou objektových tried, je však možné ho (v prístupoch odlišných od OOP) realizovať napríklad aj bežnými funkciami. Na svojej nižšej vrstve je samozrejme tvorený nejakou formou úložiska dát[1].

5.1.2. View

Interpretuje dáta z Modelu a zaisťuje grafické ale aj iné výstupy aplikácie. Cez Model pristupuje k dátam a stavom aplikácie a špecifikuje, ako majú byť prezentované. Pri zmene stavu

Modelu aktualizuje zobrazenie. Zmeny v Modeli získava buď push prístupom, kde sa len zaregistruje v Modeli a ten potom sám posiela View správy o zmenách, alebo pull prístupom, kedy sa View obracia sám na Model vždy, keď potrebuje získať najaktuálnejšie dáta. V prípade stand-alone[13] aplikácie zaisťuje View spravidla priebežné prekresľovanie a aktualizáciu obsahu zobrazených okien. Pri webových aplikáciách generuje View príslušný HTML kód, ktorý je odosielaný prehliadaču ako odpoveď na požiadavku. Ak slúži zobrazený výstup zároveň ako oblasť na zachytávanie udalostí od užívateľa,(napríklad kliknutie myšou do vykresleného okna), predáva View tieto udalosti Controlleru[1] .

5.1.3. Controller

Definuje chovanie aplikácie. Spracúva všetky vstupy a udalosti pochádzajúce od užívateľa. Na ich základe volá príslušné procesy Modelu, mení jeho stav apod. Podľa udalostí prijatých od užívateľa a podľa výsledkov akcií v Modeli potom vyberá Controller vhodné View pre ďalšie zobrazenie. U stand-alone aplikácií je takou udalosťou napríklad kliknutie myšou alebo vybranie položky menu. Pre webové aplikácie sú hlavným vstupom HTTP GET[14] alebo POST[15] požiadavky odoslané používateľovým prehliadačom[1] .

5.1.4. Koncept MVC

Hoci môže byť koncept MVC realizovaný rôznym spôsobom, všeobecne platí tento princíp:

- Užívateľ vykoná nejakú akciu v používateľskom rozhraní (napr. stlačí tlačidlo).
- Controller dostane oznámenie o tejto akcii z objektu používateľského rozhrania.
- Controller pristúpi k modelu a v prípade potreby ho aktualizuje na základe vykonanej užívateľské akcie (napr. zaktualizuje nákupný košík užívateľa).

Model je len iný názov pre doménovú vrstvu. Doménová logika spracuje zmenené dáta (napr. prepočíta celkovú cenu, dane a expedičné poplatky pre položky v košíku). Niektoré aplikácie používajú mechanizmus pre perzistentné uloženie dát (napr. databázu). To je však otázka vzťahu medzi doménovú a dátovou vrstvou, ktorá nie je architektúrou MVC pokrytá.

Komponenta pohľad použije aktualizovaného modelu pre zobrazenie zaktualizovaných dát užívateľmi. Komponenta pohľad získava dáta priamo z modelu, zatiaľ čo model nepotrebuje žiadne informácie o komponente View (je na nej nezávislý). Avšak je možné použiť návrhový vzor observer, umožňujúci modelu informovať akúkoľvek komponent o prípadných zmenách dát. V tom prípade sa komponenta View zaregistruje u modelu ako príjemca týchto informácií. Je dôležité podotknúť, že kontroler neodovzdáva doménové objekty (model) komponente pohľadu, avšak jej môže poslať príkaz, aby svoj obsah podľa modelu aktualizovala.

Samotnému konečnému zobrazeniu výsledku užívateľovi ešte môže u web-aplikácií predchádzať odpoveď zo servera na klienta, aby si ihneď vyžiadal obnovenie stránky (client side redirect, životnosť 0, takže okamžitý): Tým je zaručené, že pri obnovení stránky používateľom

13 Aplikácie schopné pracovať nezávisle na zariadení alebo systéme
14 V rámci HTTP požiadavky sa všetky formulárové dáta posielajú ako súčasť URL
15 V rámci HTTP požiadavky sa všetky formulárové dáta posielajúv tele dotazu, takže v URL nie sú vidieť.

(refresh, F5 v prehliadači) nevyvolá na serveri požadovanú akciu opakovane, ale že sa jedná iba o obnovenie pohľadu, teraz už bez požiadavky na zmenu dát (modelu). Účelom je zmena URL a dát http requestu. Celý tento client-refresh (zmena URL) sa deje automaticky a bez povšimnutia užívateľom. Užívateľské rozhranie čaká na ďalšiu akciu užívateľa, ktorá celý cyklus začne znova.

5.1.5. Hlavné prednosti MVC

- Jednoduchý prístup z rôznych druhov klientov. Pre zavedenie podpory ďalšieho klienta alebo novej role v aplikácii je potrebné nedefinovať len nové View, v prípade úplne odlišných vstupných udalostí aj špeciálny Controller, ale Model ako kľúčová časť aplikácie zostáva stále nezmenený.
- Minimalizácia duplicitného kódu. Súvisí čiastočne s predchádzajúcim bodom. Napríklad bez oddelenia a zapuzdrenia Modelu by sa pre každý nový View musela znovu programovať celá aplikačná logika. Ale i v rámci jedného typu klienta je často jedna akcia volaná z niekoľkých rôznych miest aplikácie.
- Rozdelenie vývojárskych úloh. Jednotlivé časti môžu byť vyvíjané samostatne, len so znalosťami vopred určených rozhraní medzi nimi. Minimalizuje sa dopad modifikácií, zmeny sú väčšinou vykonávané len v danej vrstve.
- Znovu použiteľnosť kódu. Ako Model je možné vo vlastnej aplikácií použiť štandardné knižnice alebo triedy. Existujú takisto univerzálne poňaté Controllery.
- Vysoká komplexnosť návrhu a jednoduchá budúca rozšíriteľnosť aplikácie, efektívna modularita.
- A v neposlednom rade aj čistota designu, spôsob premýšľania programátora nad aplikáciou,jej návrhom a architektúrou.

5.2. Hibernate

Hibernate je vlastne obdoba ORM(objektovo relačné mapovanie), je to proces pretvárania objektov do tabuľkovej reprezentácie dát. Jedným z najhlavnejších stvoriteľov projektu bol Gavin King.[2]

Hibernate sa používa k zaisteniu perzistencie dát. Perzistencia dát znamená, že dáta zostanú uchované, aj po ukončení programu. Takéto dáta majú dve dôležité vlastnosti. Po prvé, umožňujú viacnásobný prístup k dátam. Napríklad to, že dáta používa vo svojej aplikácii jeden užívateľ, nesmú byť prekážkou pre to, aby ich používali aj druhy užívateľ. Druhou z vlastností je transakčný prístup. Práve tento prístup umožňuje pracovať s dátami viac ako jednému užívateľovi, aby nedochádzalo k poškodeniu dát.

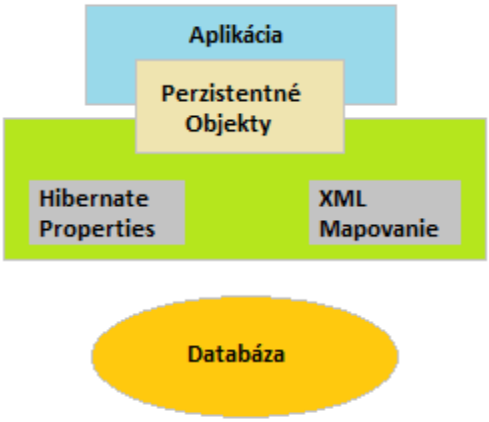
Framework Hibernate obsahuje aj vlastné dátové typy, čím uľahčuje prácu s relačnými databázami. Avšak databáza nemusí byť mapovaná len pre dátové typy Hibernatu, ale aj na základné dátové typy jazyka Java. Na tieto základné typy môžu byť dáta mapované ako hodnoty, kolekcia hodnôt alebo ako asociácia k iným inštitúciám.

Preferovaný spôsob práce s objektmi je HQL (oproti SQL). Syntax je SQL veľmi podobná, ale HQL je, na rozdiel od SQL, objektovo orientované, takže rozumie veciam ako dedičnosť, polymorfizmus, a podobne. Dotaz "from java.lang.Object o" by teda vrátil úplne všetky perzistentné objekty. Najjednoduchší dotaz vyzerá takto: "from Cennik" - vráti všetky inštancie triedy Cennik.

Ak sa v dotaze chceme odvolávať na "Cennik", je potrebné mu priradiť alias: "from Cennik as cennik" - je zvykom pomenovať aliasy rovnakým slovom ako je názov triedy. Použili by sme toho nasledujúcim spôsobom: "from Cennik as cennik where cennik.typ = typ".

HQL podporuje klasické konštrukcie, na ktoré sme zvyknutí z SQL, ako sú: inner a outer join, klauzula where alebo with, agregačné výrazy ako avg() alebo count(), subdotazov, at. Ďalšou zaujímavosťou je QBE[16]. Jedná sa spôsob dotazovanie, kedy vytvoríme nový objekt s vlastnosťami, ktoré hľadáme a tento objekt následne odovzdáme do HQL a ten nám vráti objekty s rovnakými vlastnosťami.

Hibernate je riadený konfiguračnými XML súborom, v ktorom sa predovšetkým vykonáva vlastné mapovanie objektov na tabuľky databázy.



Obrázok 5: Architektúra Hibernate

5.3. MySQL

MySQL je relačný databázový systém čo znamená, že jeho dáta sú uložené v podobe textu, čísiel alebo binárnych súborov riadených systémom správy DBMS. Keďže MySQL je databáza s voľne šíriteľným zdrojovým kódom je možné upravovať jej zdrojové kódy. Je schopná ukladať obrovské množstvo rozličných dát a informácii. MySQL je jedna z najčastejšie používaných databáz v rámci webových služieb. Je obľúbená pre podporu v rôznych programovacích jazykoch ako sú PHP,ASP ale aj Java. Práca s MySQL databázou je vykonávaná pomocou takzvaných dopytov („dotazov“), ktoré vychádzajú z programovacieho jazyka SQL (Structured Query Language).Posledná stabilná verzia nesie označenie 3.22.33 a je dostupná pre UNIX systémy aj pre Windows. Verzie s označením 3.23.XX sú momentálne uvoľnené ako alpha release.[7]

16 Query by Example

5.4. Apache TomCat

Apache Tomcat je aplikačný webový server vyvíjaný ako open-source organizáciou Apache Software Foundation. Je založený na jazyku Java, implementuje JSP (Java Servlet Pages) a EJB (Enterprise JavaBeans). Vd'aka JDK je jeho spustenie možné na operačných systémoch Linux aj Windows. Apache Tomcat vyniká jednoduchost'ou, menšou náročnosťou na výpočetný výkon na rozdiel od svojich komerčných alternatív, ktoré sú síce robustnejšie, avšak omnoho zložitejšie. Napriek svojej jednoduchosti sa hodí ako na vývoj a testovanie, tak aj na produtívnu prevádzku aplikácií. Aktuálna stabilná verzia počas písania tejto práce je 7.0.39.[8]

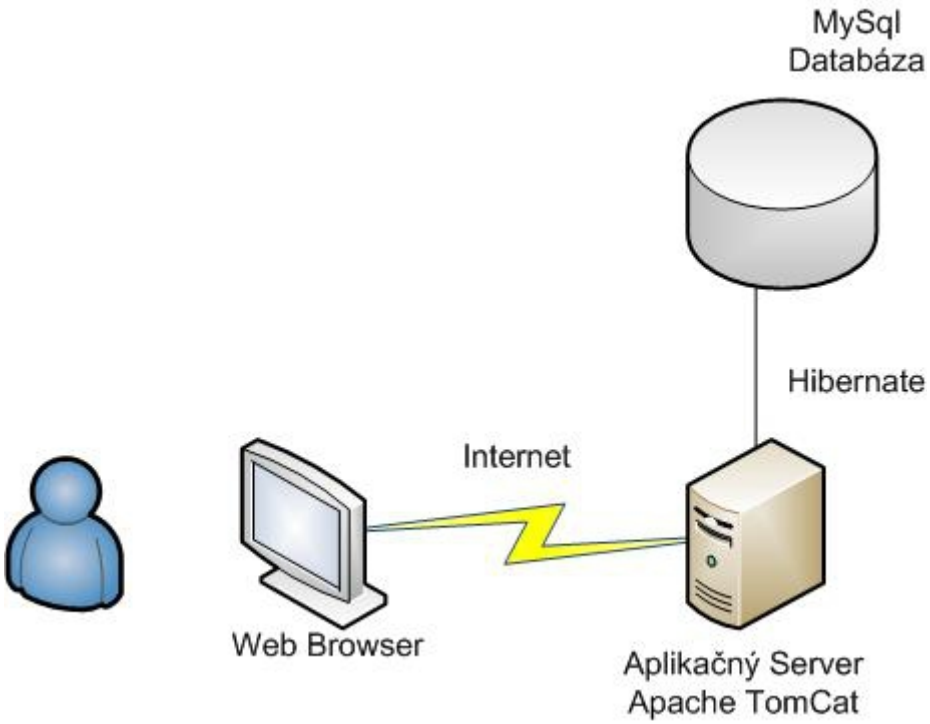
5.5. Jetty

Jedná sa o Java aplikačný webový server vyvíjaný ako časť projektu Eclipse Fundation pod licenciu open-source. Podporuje Servety, JSP stránky, Web sockety, JNDI, JMX a iné. Jednou z hlavných výhod je, že tento server dokáže pracovať v „embedded“ móde. Kvôli tejto vlastnosti je požitý v produktoch ako napríklad Apache Maven, Google app Engine, Eclipse, GWT a podobne. Jetty má verziu aj pre operačný systém android s názvom i-jetty. Otvárajú sa tak rôzne možnosti využitia aplikačného servera na mobilných zariadeniach.

6. Analýza a vlastná realizácia

6.1. Architektúra

V tejto kapitole sa budem venovať náhľadu na architektúru bližšie sa budem jednotlivými častiam venovať v samostatných kapitolách realizácie. Malý obraz na architektúru si môžete utvoriť podľa nasledujúceho obrázka.



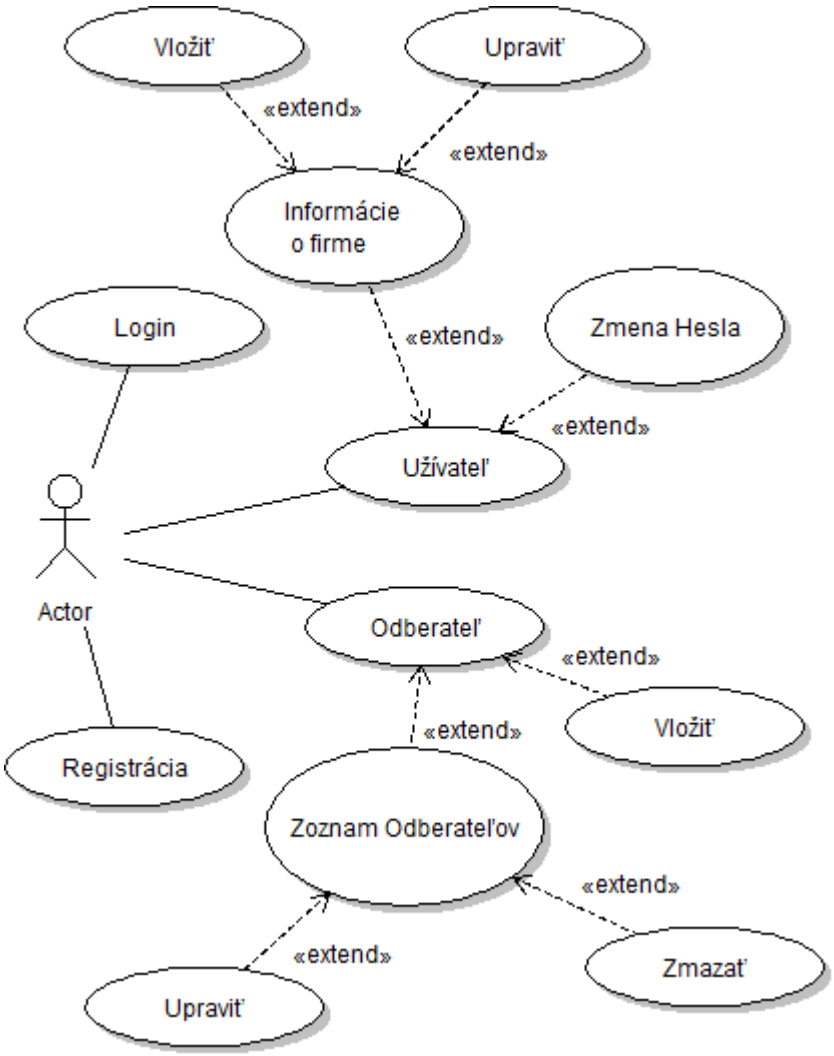
Obrázok 6: Architektúra aplikácie

Celá aplikácia je nasadená na aplikačnom serveri Apache Tomcat, ktorý je bližšie popisovaný v kapitole 5.4.. Vývoj pozostával z troch častí. Pre perzistenciu dát bola využitá MySQL databáza kapitola 5.3. Ďalšou časťou je mapovanie databázových tabuliek na java objekty. Na tento účel bolo využité ORM Hibernate popis technológie v kapitole[5.2.]. Baliček, ktorý zabezpečuje komunikáciu s databázou môžeme pomenovať business vrstva. Je samostatný projekt uložený v súbore typu jar. V tejto vrstve sa nachádzajú dva dôležité baličky „bo“ a „dao“. V „bo“ baličku sa nachádzajúcu rovnaké objekty ako v databáze, ktoré sú pomocou anotácii presne mapované na tabuľky v databáze. V baličku Dao sa nachádzajú metódy, ktoré slúžia priamo na operácie tabuľkami v databáze, ako je napríklad ukladanie, update načítavanie a iné. Ďalšou časťou sú ovládacie prvky, takzvané controllery, tie majú na starosti chovanie UI dotvárajú vzhľad a určujú kedy a s akými informáciami sa má daný komponent zobraziť. Ako posledné sú pomocne utility, ktoré zabezpečujú radu funkcii aplikácie.

6.2. Analýza

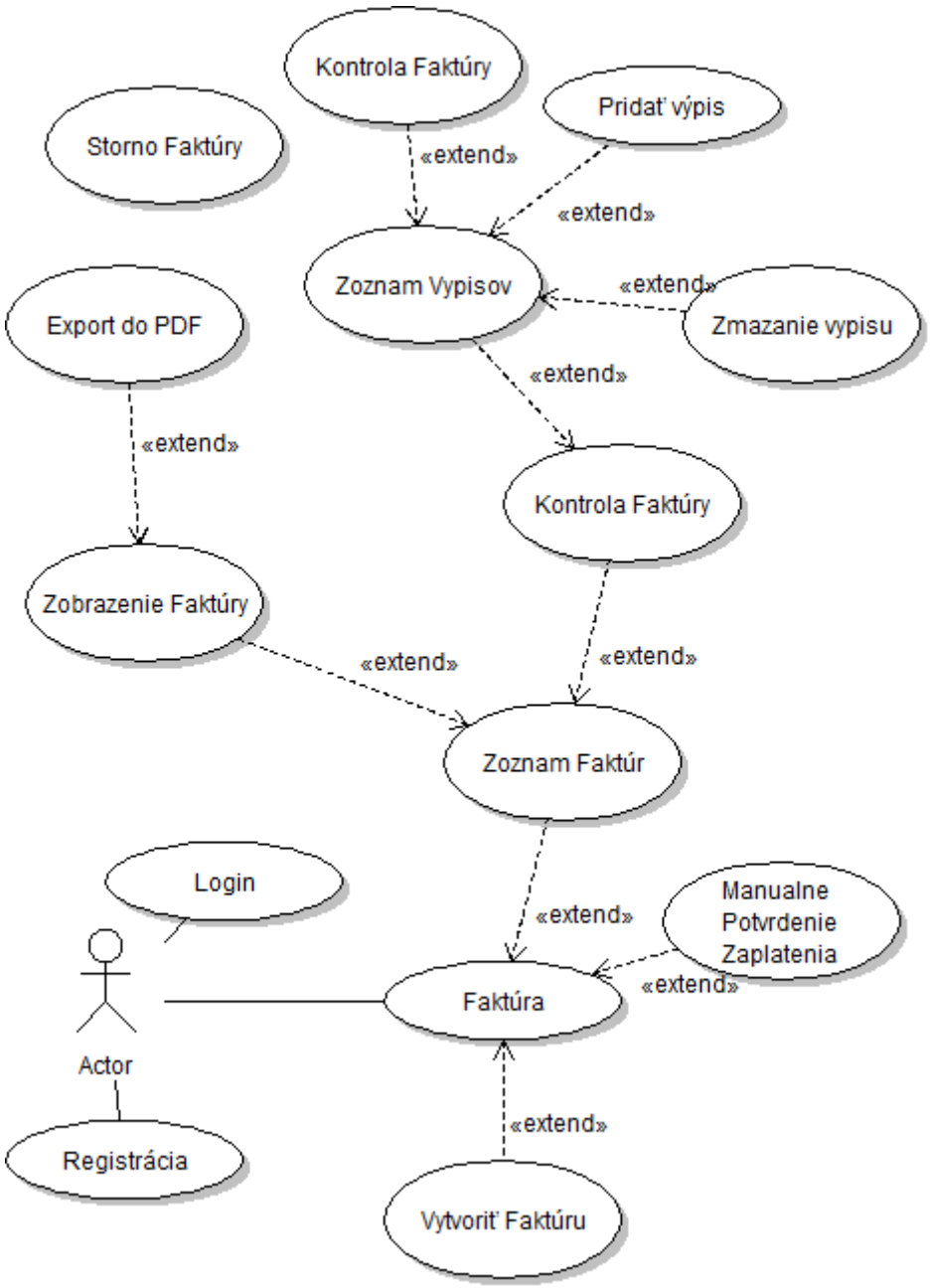
6.2.1. UseCase modely

- Use Case – model prípadov užitia, znázorňuje popis a použitie funkcií systému z pohľadu užívateľa pri voľbe Užívateľ a Odberateľ Obrázok 7: UseCase Užívate a Odberateľ.



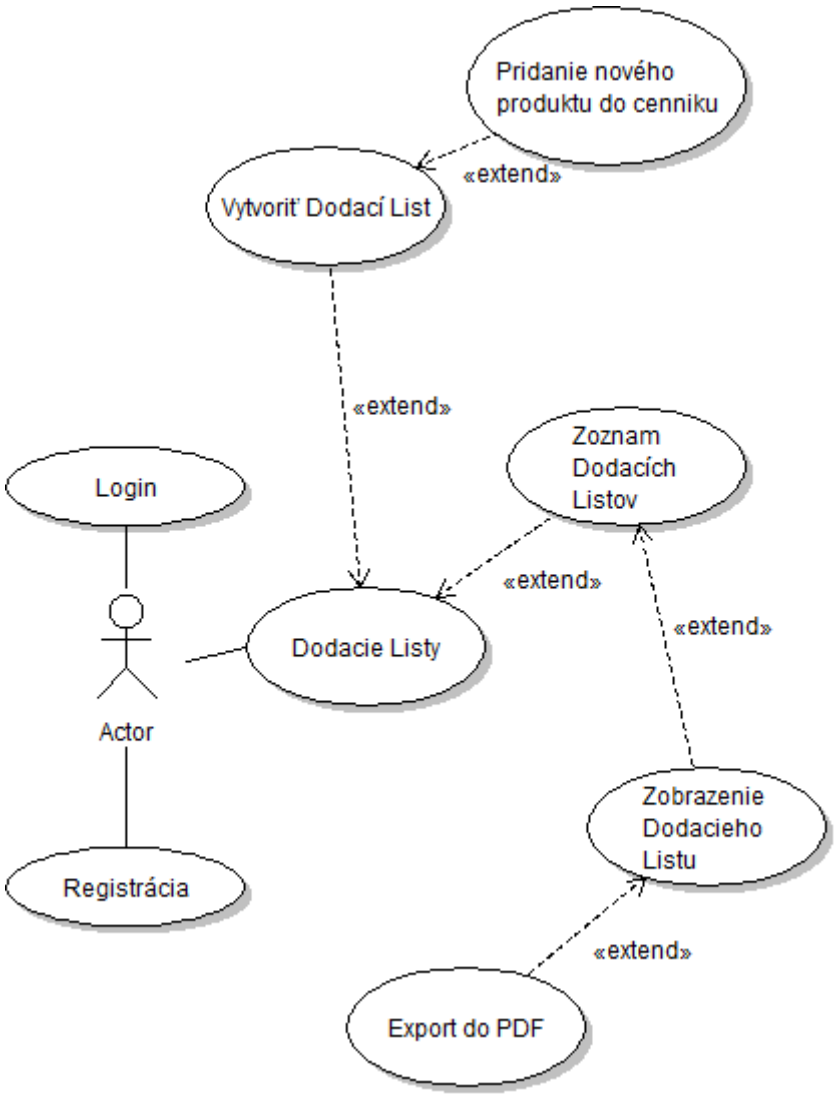
Obrázok 7: UseCase Užívate a Odberateľ

- Use Case – model prípadov užiťia, znázorňuje popis a použitie funkcií systému z pohľadu užívateľa pri voľbe Faktúra Obrázok 8: UseCase Faktúra.



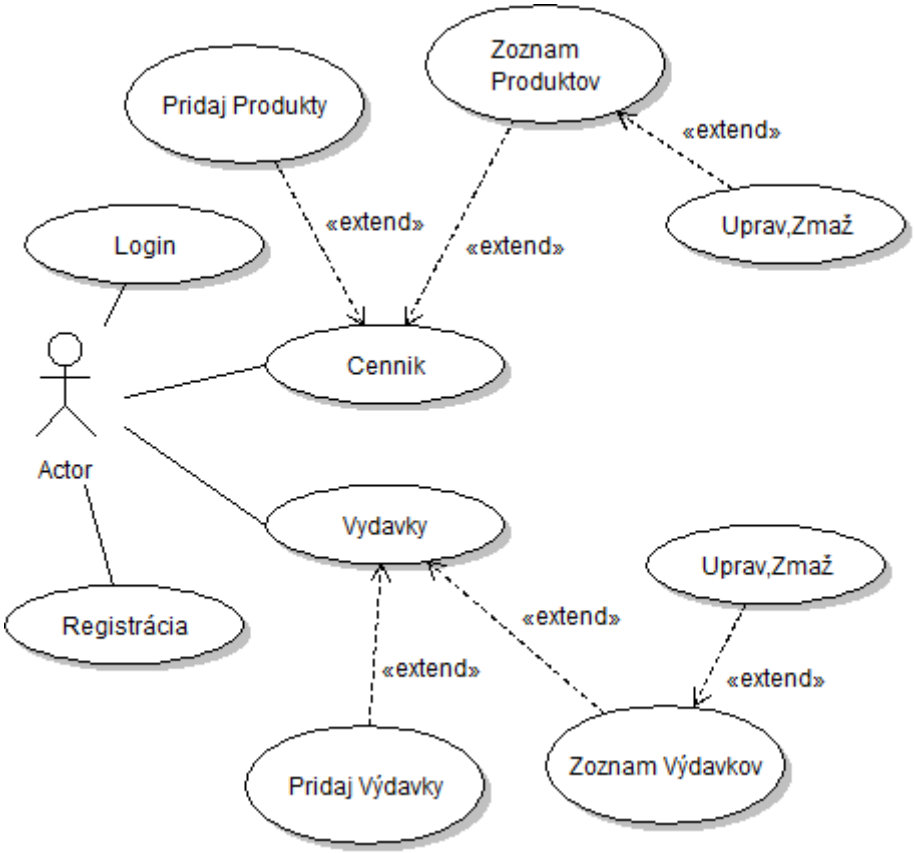
Obrázok 8: UseCase Faktúra

- Use Case – model prípadov užiťia, znázorňuje popis a použitie funkcií systému z pohľadu užívateľa pri voľbe Dodacie Listy Obrázok 9: UseCase Dodací List.



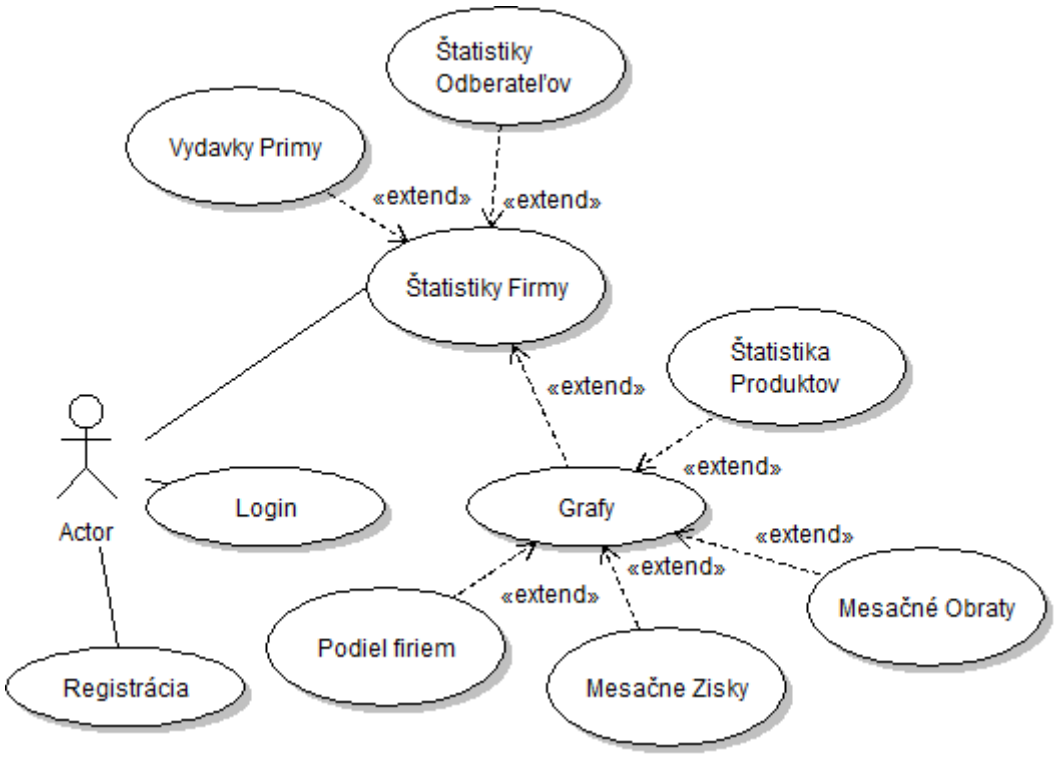
Obrázok 9: UseCase Dodací List

- Use Case – model prípadov užiťia, znázorňuje popis a použitie funkcií systému z pohľadu užívateľa pri voľbe Cennik a Výdavky Obrázok 10: UseCase Cennik a Výdavky.



Obrázok 10: UseCase Cennik a Výdavky

- Use Case – model prípadov užiťia, znázorňuje popis a použitie funkcií systému z pohľadu užívateľa pri voľbe Štatistiky Firmy Obrázok 11: UseCase Štatistky Firmy.



Obrázok 11: UseCase Štatistky Firmy

6.3. Tvorba MySql Databázy

Dôvody pre implementáciu MySQL:

- nenáročná na hardware,
- nie je zbytočne zložitá ako napríklad Oracle, jej možnosti viac ako postačujú pre tento typ aplikácie,
- výborná dokumentácia,
- veľa utilít na uľahčenie tvorby databázovej štruktúry,
- open-source možnosti,
- v neposlednom rade cena.

6.3.1. Databáza

Ešte pred začiatkom samotnej tvorby entít v databáze sa museli použiť dve najlepšie pomôcky programátora papier a pero. Po niekoľkých úvodných náčrtoch nakoniec uzrela svetlo sveta databáza fakturačného systému, ktorá obsahovala nasledovné entity :

- **mesto** - nachádzajú sa tu názvy všetkých slovenských miest a obcí spolu psč, keďže za podobná databáza môže stáť viac ako 130 €, bolo nutné si ju vyrobiť. Použitý bol textový súbor, ktorý obsahoval názov mesta a príslušné psč. Jednoduchou metódou za pomoci načítania z textového súbora a hibernate príkazov bola databáza vytvorená.
- **dodaci** - dodacie listy užívateľa,
- **dodaciprodukty** - sú tu uložené produkty dodacích listov,
- **faktura** - faktúry užívateľa,
- **login** - prihlasovacie údaje užívateľa,
- **uzivatel** - informácie o užívateľovi,
- **cennik** - zoznam produktov užívateľa,
- **vydavky** - zoznam výdavkov užívateľa,
- **odberatel** - údaje o odberateľoch.

Podrobná štruktúra entít a ich vzťahy môžete vidieť v ER Diagram č.1.

Entity boli vytovrene v MySQLAdministratorovi. V tomto programe je možná aj záloha celej schémy v MySQL skripte. Pre vývoj musel byt nainštalovaný aplikačný server ako na príklad v tomto prípade Apache TomCat popisovaný v kapitole5.4..

6.3.2. Business vrstva

Vytvorenie bussines vrstvy pozostávalo z dvoch častí. V prvej časti vytvorenie java objektov, ktoré sú v balíčku `net.faktura.bus.bo` tieto objekty sú presným obrazom entít v databáze. Pomocou Hibernate a potrebných anotácií, ktoré sa nachádzajú v balíčku `javax.persistence`. Anotácie sú rozdelené do viacerých typov.

Typy anotácií sú nasledovné :

- na previazanie triedy s entitou tabuľky kód uvedený nižšie

```
@Entity
@Table(name = "cennik")
public class Cennik
```

- previazanie ID entitiy ako generovaneho atributu kód uvedený nižšie

```
@Id
@GeneratedValue
@Column(name = "id_cennik")
private int id;
```

- previazanie stĺpca s atribútom triedy kód uvedený nižšie

```
@Column(name = "typ")
private String typ;
```

- previazanie medzi dvoma tabuľkami pomocou cudzieho :

1. väzba N:1 kód uvedený nižšie

```
@@ManyToOne
@JoinColumn(name = "login")
private Login login;
```

2. väzba 1:N kód uvedený nižšie

```
@OneToMany(mappedBy = "dodaci", cascade = CascadeType.ALL)
private Set<DodaciProdukty> dodaciProdukty;
```

3. väzba 1:1 kód uvedený nižšie

```
@OneToOne
@JoinColumn(name = "id_sklad")
private Sklad sklad;
```

Po previazaní jednotlivých atribútov tried, museli byt vytvorene set a get metódy. Metoda set slúži na nastavenie atribútu a metóda get na vrátenie jeho nastavenej hodnoty tieto metódy znázornuje nižšie uvedený kód.

```
public int getId() {
```

```
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}
```

Za posledné bolo potrebné nastavenie hibernatu inak by nám naše úsilie bolo na nič a to za pomoci konfiguračného hibernate.cfg.xml súboru ktorý sa nachádza v samostatnom balíčku `net.faktura.bus.config`. V nasledujúcom kóde je ukážka xml súbora už s prevedenými nastaveniami bez ktorých by sa hibernate nedokázal spojiť s MySQL databázou.

```
<hibernate-configuration>
	<session-factory>
		<property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</pr
operty>
		<property
name="hibernate.connection.username">dzefri</property>
		<property
name="hibernate.connection.password">8891</property>
		<property
name="hibernate.connection.url">jdbc:mysql://localhost/fakturacna<
/property>
		<property
name="hibernate.default_schema">fakturacna</property>
		<property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prope
rty>
		<property
name="hibernate.connection.characterEncoding">utf8</property>
		<property name="connection.pool_size">1</property>
		<property
name="hibernate.hbm2ddl.auto">update</property>
		<property
name="hibernate.connection.shutdown">true</property>
		<property name="show_sql">false</property>

		<!-- C3P0 Connection Pool -->
		<property name="c3p0.acquire_increment">5</property>
		<property name="c3p0.idle_test_period">3600</property>
 <!-- seconds -->
		<property name="c3p0.min_size">5</property>
		<property name="c3p0.max_size">20</property>
		<property name="c3p0.timeout">1800</property>
<property name="hibernate.c3p0.max_statements">50</property>
<property name="c3p0.testConnectionOnCheckout">true</property>
<property name="current_session_context_class">thread</property>
<!-- Disable second-level cache. -->
<property
```

```
name="cache.provider_class">org.hibernate.cache.EhCacheProvider</p
roperty>
<property name="cache.use_query_cache">false</property>
<property name="cache.use_minimal_puts">false</property>
          <property name="max_fetch_depth">3</property>


          <mapping class="net.faktura.bus.bo.Sklad" />
          <mapping class="net.faktura.bus.bo.Cennik" />
          <mapping class="net.faktura.bus.bo.Zamestnanec" />
          <mapping class="net.faktura.bus.bo.Odberatel" />
          <mapping class="net.faktura.bus.bo.Faktura" />
          <mapping class="net.faktura.bus.bo.Dodaci" />
          <mapping class="net.faktura.bus.bo.Uzivatel" />
          <mapping class="net.faktura.bus.bo.Vydavky" />
          <mapping class="net.faktura.bus.bo.Login" />
          <mapping class="net.faktura.bus.bo.Mesto" />
          <mapping class="net.faktura.bus.bo.DodaciProdukty" />
     </session-factory>
</hibernate-configuration>
```

V druhej časti vytvorenie tried, ktoré sú uložené v balíčku **package** `net.faktura.bus.dao` (ďalej len Dao). Tieto triedy obsahujú metódy, v ktorých sa vykonávajú transakcie.

Prvý krok v metódach je vždy získanie Hibernate Session objektu, ktorý tvorí hlavný interface medzi Java aplikáciou a hibernate. SessionFactory umožňuje získanie Session pomocou údajov v konfiguračných súboroch. Po vytvorení bloku transakcie aplikácia pomocou session adekvátnym spôsobom použije perzistentný objekt, čo sa zvyčajne odrazí na obsahu zodpovedajúcej databázovej tabuľky. Každá z tried obsahuje niekoľko takýchto metód s transakciami, v ktorých sa nachádzajú dotazy na databázu, sú vytvorené criteriami, pomocou týchto dotazov vykonávame potrebné operácie s databázou v kóde pod textom sa nachádza kompletná metóda z balíčka Dao. Ostatné metódy sú vytvorené podobne nachádzajú sa v nich iba iné dotazy typov ako save, saveOrUpdate, delte a ďalšie. Na vyberanie špecifickych stĺpcov z tabuliek sú využité taktiež criteria.

```
     public List<Cennik> findByLogin(Login login) {

          List<Cennik> objects = null;
          Transaction transaction = null;

          Session session =
HibernateUtil.getSessionFactory().openSession();
          try {
               transaction = session.beginTransaction();
               Criteria criteria =
session.createCriteria(Cennik.class);
               criteria.add(Restrictions.eq("login", login));
               objects = criteria.list();
               transaction.commit();
          } catch (HibernateException e) {
```

```
                e.printStackTrace();
                System.out.println(e.getMessage());

            } finally {
                session.close();
            }
            return objects;
        }
```

6.4. Logika

Vyvíjaná bola v prostredí Eclipse, do ktorého bol vložený plugin ZK, ten obsahoval novú perspektívu ktorej obsahom bol ZUL designer, preview ZUL formulárov a hlavne API ZK. Logika aplikácie je rozdelená do troch balíčkov :

- net.faktura.ui.controllers,
- net.faktura.ui.utils,
- net.faktura.ui.grafy,
- net.faktura.ui.zk.

Vybrané triedy sú popisované v Triednych diagramoch : 2: ClassDiagram 3: ClassDiagram 4: ClassDiagram

6.4.1. Balíček controllers

Každý ZUL súbor je skonvertovaný do HTML a JavaScriptu, reprezentuje View, respektíve vizuálnu časť aplikácie, na jej pozadí prebiehajú rôzne procesy, ktoré zaisťujú interaktivitu a vykonanie užívateľských požiadaviek.

ZUL obsahuje komponenty ako napríklad button, textbox, grid, listbox a podobne. Aby sme mohli tieto komponenty ovládať pomocou controllerov tak sa musí najskôr previazať samotný controller so ZUL súborom a to tak ze si zvolíme hlavný komponent ZUL súbora a do atribútu apply uvedieme presnú cestu k danému controlleru. Ako hlavný komponent môže byť napríklad <window>, potom by konfigurácia vyzerala nasledovne kódom uvedeným nižšie :

```
<window
apply="net.faktura.ui.controllers.PridajProduktCennikController">
```

Ďalej musí byť nakonfigurovaný Controller, ktorý je v podstate java trieda a tá musí dediť z triedy SelectorComposer<Component>, v tomto prípade direktíva <Window>, tato trieda sa nachadza v balíčkoch API ZK. Na previazanie komponentov s controllerom je potrebné unikátne id komponentu v ZUL napriklad `<button id="plusBtn" />` a v controllery vytvorený atribút daného typu komponentu za jeho názov musí byť zvolené id komponentu a nad touto deklaráciou musí byť uvedená kľúčová anotácia wire. Ukážka previazania atribútu z pohľadu controllera nasledujúci kód:

```
@Wire
private Button minusBtn;
```

Po previazaní môžeme v metódach meniť vlastnosti respektíve atribúty komponentu, napríklad jeho visibilita, veľkosť, rozloženie a podobne.

Controllery nie lenže ovládajú komponenty, ale na pozadí pracujú s dátami a vykonávajú požiadavky ktoré im prostredníctvom UI zadáva užívateľ, posielajú signály bussines vrstve na prevzatie dat alebo naopak ich vloženie, update alebo delete. Pre spojenie s bussines objektami je potrebné importovať balíčky s objektami, ktoré v danom kontroleri používame, ukážka importu kód uvedený nižšie.

```
import net.faktura.bus.bo.Login;

import net.faktura.bus.dao.CennikDao;
```

6.4.2. Balíček utils

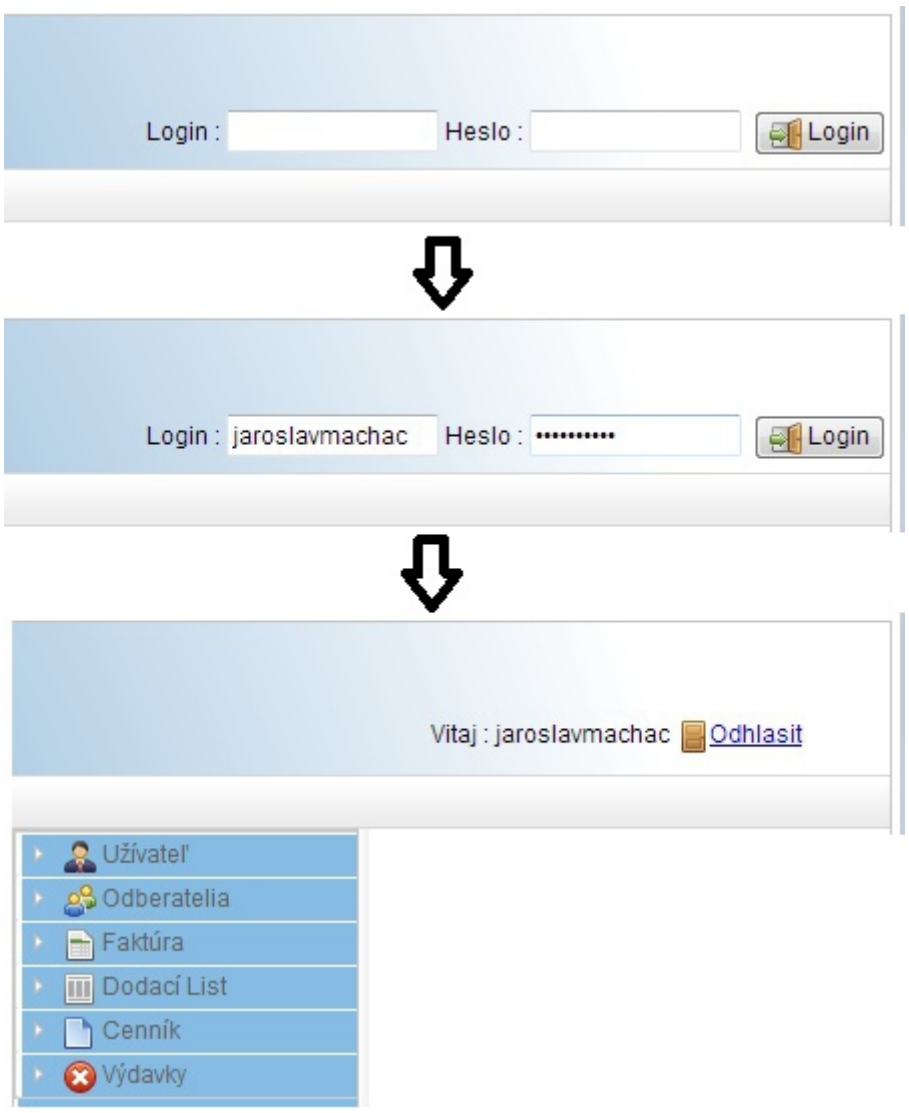
V tomto balíčku sa nachádzajú triedy, ktoré pomáhajú pri chode aplikácie a predlohy pdf súborov, ktoré sú potrebné na export dodacích listov a faktúr.

Podporné triedy sú nasledovné :

- EventHelper
- ExportPdfDodaci
- ExportPdfFaktura
- KontrolaFaktury
- SessionDataHolder
- VydavkyPrimy
- LoginCredentialManager

6.4.2.1. EventHelper

Táto trieda rieši problematiku komunikácie medzi controllermi. Controller, ktorý chce byť notifikovaný o akciách, musí implementovať rozhranie EventHelperListener. Tento controller musí byt zaregistrovaný v triede EventHelper resp. táto trieda obsahuje zoznam inštancií typu EventHelperListener. Pri vzniku akcie sa následne tieto inštancie iterujú za účelom zavolania metódy so vstupným parametrom, ktorý identifikuje danú akciu. Notifikovaný controller môže následne na takúto akciu reagovať vlastným chovaním, ako je napríklad zobrazenie stromového menu po prihlásení užívateľa. obrázok č.12.



Obrázok 12: Ukážka fungovania EventHelperu s pohľadu View

6.4.2.2. ExportPdf

Dôležitou častou aplikácie je tlač a náhľad na dokumenty, preto sa v aplikácii nachádzajúcu triedy, ktoré umožňujú export do formátu PDF. V týchto triedach sú metódy, ktoré obsahujú atribúty. Do vopred pripravených formulárov, ktoré boli vyhotovené v trial verzií programu Adobe LiveCycle designer ES, tieto formuláre sú vo formáte PDF, v ktorých sú komponenty s unikátnym „id“, tieto komponenty je možné previazať s Java triedou pomocou API Itext[17]. Po previazaní je aplikácia schopná naplniť tieto komponenty potrebnými dátami. Boli vytvorené dve verzie takýchto formulárov pre export faktúry a export dodacieho listu sa nachádzajú na CD v PDF.

17 open source API pre tvorbu a manipuláciu s PDF súbormi v Jave

6.4.2.3. KontrolaFaktury

Touto triedou sa aplikácia odlišuje od existujúcich riešení. Umožňuje používateľovi kontrolovať zaplatenie faktúry za pomoci variabilného symbolu a výpisu z firemného bankového účtu. Užívateľ nahrá na server svoj výpis z banky, následne po tom je možné previesť kontrolu o zaplatení. Kontrola prebieha tak ze metóda prechádza riadky zo strán PDF výpisu a hľadá zhodu variabilného symbolu, po nájdení zhody užívateľ dostane hlášku o zhode, v ktorej sa vypíšu aj tri riadky z výpisu aby sa užívateľ uistil či je táto zhoda platná, potom ako sa týmto výpisom užívateľ uistí že je to správna úhrada tak môže zvoliť potvrdenie alebo naopak keď si nie je istý môže zvoliť voľbu cancel, ktorá ho vráti späť na zoznam výpisov bez zmeny stavu faktúry, ale ak si zvolí potvrdenie, tak sa faktúra prevedie do potvrdeného stavu. Ak metóda nenájde zhodu informuje o tom užívateľ informatívnou hláškou, že sa zhoda v danom výpise nenašla.

6.4.2.4. SessionDataHolder

Je to v podstate zásobník dát, ktoré využíva aplikácia, tieto dáta sú ukladané na session užívateľa, ktorú dostáva užívateľ pri otvorení aplikácie. Týmito dátami sú aj dáta z databázy, sú ukladané jednotlivo alebo do generických kolekcií. Po uložení dát z databázy do SessionDataHoldera, sa aplikácia stáva nezávislou na databáze a nemusí zbytočne zaťažovať databázu neustálim dokazovaním na dáta. Pri zmene dát sa aktualizujú dáta ako v databáze tak aj v SessionDataHoldery. Logicky táto vystupuje ako Singlton jedinečný pre session. Ukážka vytvorenia singletona a atribútov kód č.[]

```
private static final String KEY_eventHelper =
EventHelper.class.getName();

      HashMap<String, EventHelperListener> listHelperListeners;

      public static EventHelper getInstance() {

            return getInstance(Sessions.getCurrent());

      }

      public static EventHelper getInstance(Session zkSession)

      {
            synchronized (zkSession) {
                  EventHelper helperModel = (EventHelper) zkSession
                              .getAttribute(KEY_eventHelper);
                  if (helperModel == null) {
                        zkSession.setAttribute(KEY_eventHelper,
                                    helperModel = new EventHelper());
                  }
                  return helperModel;

            }
```

6.4.2.5. VydavkyPrimiy

Tato metóda implementuje rozhranie `Comparable<VydavkyPrimy>`, slúži na porovnávanie a zatrieďovanie dvoch rôznych objektov typu výdavky a typu faktúry. Obsahuje atribúty :

- typ string,
- datum date,
- suma double,

ktoré získava z týchto objektov.

6.4.2.6. LoginCredentialManager

Rieši problematiku autentifikácie, taktiež je vytvorena podla návrhového vzoru Singleton. Ak sa uzivatel pokúsi prihlásiť zavolá sa metóda login(), ktorá ma dva vstupy typu string a to sú meno a heslo, metóda vykoná dotaz na databázu, z ktorej získa generickú kolekciu lsit typu login, ktorú následne na to iteruje a ak nájde zhodu s oboma vstupnými parametrami prihlási užívateľa. Metóda isAuthenticated() slúži na overenie prihlásenia vracia hodnotu typu boolean, ak sa vrátená hodnota rovná true znamená to ze uzivatel je uspesne prihlaseny a moze pokracovat v pouzivani aplikacie. Prihalasenie zanika dvomi sposobmi bud zvolenim volby logoff, kedy sa zavolá metóda logoff() ktorá na instancii credetialManagera priradi atributu login hodnotu false alebo pri zaniknuti session. Kód č.[] su znazornene metody login a logoff triedy LoginCredentialManager.

```
public synchronized void login(String meno, String password) {
            Login tempLogin = loginDao.findLogin(meno);
            if (tempLogin != null &&
tempLogin.getPass().equals(password)) {
                  login = tempLogin;
            } else {
                  login = null;
            }
      }

public synchronized void logOff() {
            login = null;
      }
```

6.4.3. Grafy

Balíček bol vytvorený na sprehľadnenie architektúry. Nachádzajú sa v ňom controllery, ktoré ovládajú View grafov a štatistik firmy užívateľa.

6.5. View

View sa skladá s troch hlavných častí, vrchná kde sa nachádzajú polia pre prihlasenie do aplikácie, tlačítko pre registráciu a informácie. V informáciach je používateľský manuál. Po prihlásení pribudne v hornej lište drop menu, kde sa nachádzajú štatistiky firmy a druhá hlavná časť stromové menu kde sa nachádzajú všetky podstatné funkcie aplikácie. Poslednou časťou je

content v nom sa zobrazujú všetky údaje a výsledky, ktoré si užívateľ vyžiada pri používaní aplikácie.

6.6. Nasadenie IS na aplikačný server Apache TomCat

Aby sme boli schopný nasadiť aplikáciu na takýto server musíme ju z vývojového prostredia najskôr exportovať vo formáte war[18] v Eclipse to dosiahneme kliknutím v hlavnom menu projektu na export, kde vyberieme z možností export do war súboru. Ďalším bodom je deploy(nahranie) tohto súboru na aplikačný server pomocou interface, ktorý poskytuje Apache TomCat, tento interface si otvoríme zadaním správnej adresy servera do web browsera. Vyberieme možnosť deploy war file a následne na to vyberieme súbor, ktorý sme exportovali niekam na lokálny disk. Ďalším krokom je upload Bussines vrstvy ktorá nám zabezpečuje komunikáciu s databázou. Ale najskôr si musíme nastaviť konfiguráciu pre daný server a databázu a to pomocou konfiguračného xml súboru ukážka takéhoto nastavenia je v kapitole[] . Po nastavení konfiguračného súboru obdobným spôsobom, ako súbor var exportujeme Bussines vrstvu, ale do formátu jar. Tento balíček potom nahráme medzi hlavné balíky Apache TomCat. Ďalším krokom je pomocou sql scriptu a MySQL Administrátora vytvoriť na servri databázu, na ktorú už máme vopred nakonfigurovanú Bussines vrstvu.

Po týchto úkonoch je aplikácia pripravená na použitie stačí len do web browseru zadať správnu adresu a aplikácia sa spustí.

18 Web application ARchive - JAR súbor používaný na distribúciu zbierky JSP, Java servlety, Java triedy, súbory XML, balíčky, statické webové stránky (HTML a súvisiace súbory) a ďalšie zdroje, ktoré spolu predstavujú webovú aplikáciu.

7. Záver a zhodnotenie práce

Cieľom práce bolo vyvinúť jednoduchý IS, ktorý rieši problematiku fakturácie. Spracovanie dát ich vizualizáciu pre potreby užívateľa.

Pred začatím prace som si musel naštudovať problematiku fakturácie, aké náležitosti majú mať dokumenty podla legislatívy spojené s touto činnosťou ako sú dodacie listy a faktúry. Čo všetko je pre užívateľa tohto systému dôležité, aké dáta a štatistické údaje rozhodujú, a v prvom uľahčujú orientáciu vo vlastných fakturačných dátach, ktorá nám šetrí drahocenný čas. Aplikáciu som sa snažil navrhnúť veľmi intuitívne, aby ovládanie bolo zrozumiteľne a jasné aj pre manej zdatných v oblasti informačných technológií, ktoré v dnešnej dobe tak úžasne rýchlo napredujú. Na mojom ocovi som si všimol, že každá mala návesť mu dokáže lepšie sa zorientovať napríklad v systéme Windows. Na tomto základe som postavil celú aplikáciu,v ktorej sa nachádzajú informačné tooltipy ,ktoré objasňujú chovanie jednotlivých komponentov.

S týmito poznatkami som sa pustil do samotnej implementácie aplikácie, ktorá sa skladá s troch základných častí databázy, logiky a Viewu.

Pri implementácii som si musel ozrejmiť technológie a návrhové vzory, ktoré sú potrebné pre vytvorenie aplikácie tohto typu. Aplikácia bola vyvíjaná vo webovom aplikačnom frameworku ZK. Pre je ho správne využitie som si musel naštudovať jeho API a prejsť radu tutoriálov. V databázovej časti sa jednalo vytvorenie databázy pre zaistenie perzistencie dát, zaistenie jej ORM pomocou Hibernate a Bussines vrstvy. Pri implementácií logiky to bol jazyk Java a OOP(Objektovo orientovane programovanie). Obsahovala množstvo tried, ktoré zaisťovali funkcie ako napríklad kontrola bankových vypisov, vytváranie interaktívneho prostredia tvorbu, dynamických ZUL komponentov, ktoré sú súčasťou ZK frameworku a podobne. Pri tvorbe View som postupoval citlivo dbal som na návrh menu a jednotlivých komponentov aby nepôsobili zbytočný zmätok. Nakoniec som aplikáciu nasadil na aplikačný server Apache TomCat.

Vyhliadky na zdokonalenie a rozšírenie. Na radu vylepšení som prišiel už počas vyvoja no pre nedostatok času sa mi ich nepodarilo implementovať. Spomeniem niektoré s týchto napadol na zlepšenie :

- vytvorenie aplikácie pre smártfóny pomocou, ktorej by bolo možné kontrolovať stav fakturácie, vytváranie dokumentov a celkové ovládanie,
- skladovú politiku,
- vytvorenia apy aby bolo možne systém zosynchronizovať napríklad s e-shopom,
- vytvorenie peňažného denníka, pomocou ktorého by bolo možné exportovať daňové priznanie,
- správa zamestnancov.

Dá sa povedať, že zadanie tejto bakalárskej práce som splnil v celom rozsahu. Po celkovej realizácií sa stala táto práca pre mňa jednoznačne prínosom z hľadiska ozrejmenia rôznych technológií a v poukázaní na možnosti využitia webovo aplikačného frameworku ZK na poli online fakturácie.

8. Literatúra

[1] PECINOVSKÝ, Rudolf. Návrhové vzory - 33 vzorových postupů pro objektové programování. Brno : Computer press, 2007.

[2] Hibernate [online]. Hibernate Reference Dokumentation. Dostupné z: http://hibernate.org/docs

[3] ZkFramework [online]. Zkoss Documentation Dostupné z : http://www.zkoss.org/documentation/

[4] Apache Tomcat 7 Documentation [online]. Dostupné z: http://tomcat.apache.org/tomcat-7.0-doc/index.html

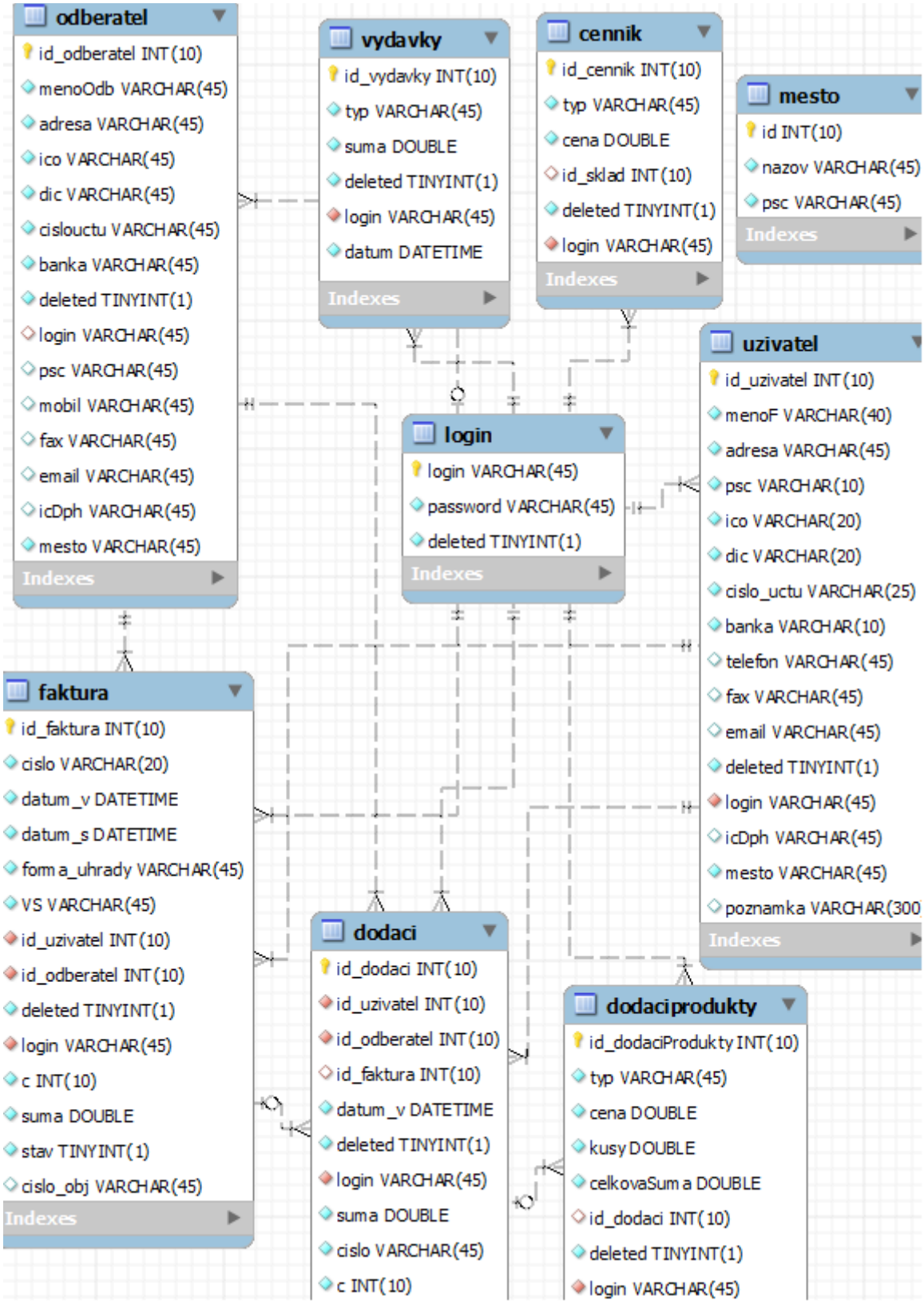
[5] Metodic ký pokyn k fakturácii podľa zákona č. 222/2004 Z. z. o dani z pridanej hodnoty v znení neskorších predpisov[online]. Dostupný z : http://www.drsr.sk/drsr/slovak/danovy_subjekt/pokyny_dr_sr/data/mp_2012_11_05_fakturacia.pdf

[6] Predpis č. 431/2002 Z. z. Zákon o účtovníctve [online] dostupný z : http://www.szk.sk/files/legislativa/2002-431_znenie_20120101.pdf

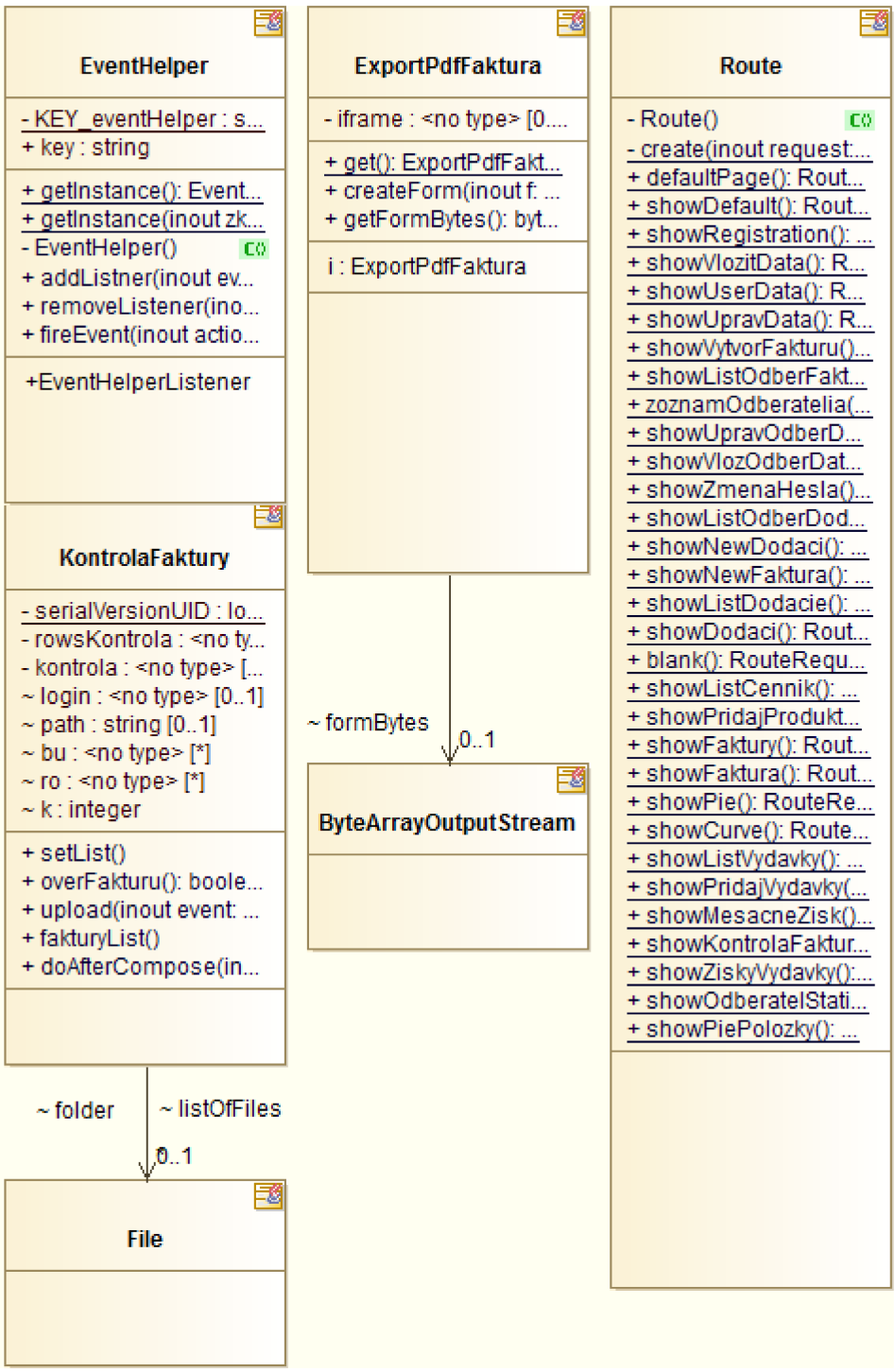
[7] Gilmore, Jason W. Beginning PHP and MySQL: From Novice to Professional. Berkeley : Apress, 2008. str. 1080. ISBN 978-1590598627.

[8] Google Web Toolkit. Developer's Guide[online] dosttupný z : https://developers.google.com/web-toolkit/doc/latest/DevGuide

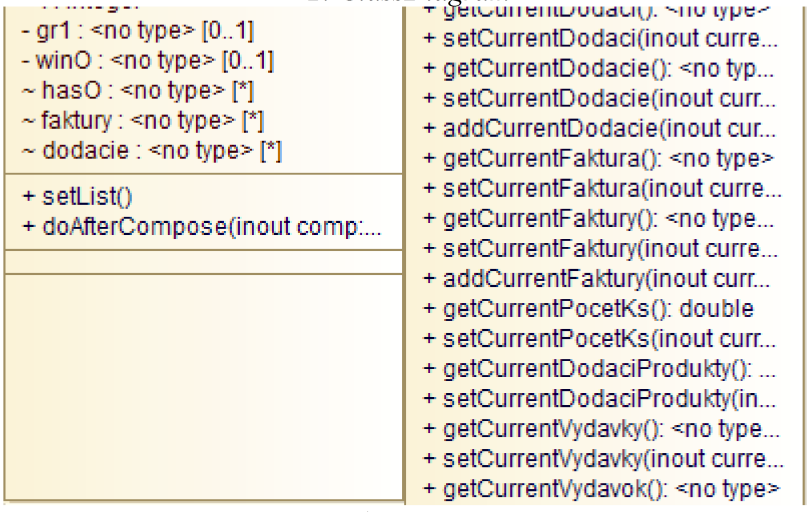
9. Prílohy



ER Diagram č.1



2: ClassDiagram



3: ClassDiagram

4: ClassDiagram